Vorhersage der Datenübertragungsraten und eNodeB-Verbindungsdauern in LTE-Netzen

Christian Peters

5. Januar 2021

Veranstaltung:	Fallstudien II
Dozent:	Prof. Dr. Markus Pauly
Gruppe:	Laura Kampmann, Christian Peters, Alina Stammen

Inhaltsverzeichnis

1 Einleitung

Für neuartige Verkehrstechnologien, wie das autonome Fahren, sind zuverlässige Kommunikationsmechanismen zu den Mobilfunknetzen von besonders großer Bedeutung [5]. Um die immer weiter steigenden Anforderungen an die Verbindungsqualität einhalten zu können, bedarf es der kontinuierlichen Weiterentwicklung immer effizienterer Kommunikationsverfahren zwischen Endgerät und Netzwerk.

Bevor solche Verfahren jedoch in der Praxis eingesetzt werden können, bedarf es zunächst ausgiebiger Tests. Dies stellt die Wissenschaftler allerdings häufig vor ein Dilemma: Während die aussagekräftigsten Ergebnisse natürlich immer in einer realen Messumgebung erzielt werden können, sind derartige Experimente meist sehr aufwändig, kostspielig und auch nur begrenzt reproduzierbar. Oft kommen daher als Alternative sogenannte Netzwerksimulationen zum Einsatz, welche diese Probleme zwar lösen, aber dafür aufgrund der zahlreichen Vereinfachungen weniger aussagekräftige Ergebnisse liefern können.

Um diese Schwachstellen zu beheben, wurde in [5] ein neues Simulationsverfahren vorgestellt, welches die Aussagekraft realer Experimente mit den Vorzügen der Netzwerksimulation vereinen soll. Es handelt sich hierbei um die sogenannte Data-driven Network Simulation (DDNS), welche das Verhalten von Netzwerken basierend auf real erhobenen Netzdaten simuliert, um so die Aussagekraft von Simulationen zu steigern.

Ein zentraler Baustein innerhalb von DDNS ist dabei die möglichst realitätsnahe Prognose von Datenübertragungsraten basierend auf anderen real gemessenen Netzwerkindikatoren. Zu diesem Zweck kommen bei DDNS Prädiktionsverfahren des Machine Learning zum Einsatz. Man erhofft sich durch das Training dieser Verfahren auf realen Netzwerkdaten eine möglichst realistische Prognose der Datenraten und damit auch möglichst aussagekräftige Simulationsergebnisse.

In diesem Projekt soll nun einmal die Eignung zweier Prognoseverfahren, *Extreme Gradient Boosting* und *Lineare Regression mit ARMA-Fehlern* anhand real erhobener Netzwerkdaten der Anbieter O2, T-Mobile und Vodafone im Raum Dortmund untersucht werden. Zu diesem Zweck werden die Modelle für jeden der Anbieter getrennt angepasst und anschließend auf ungesehenen Daten ausgewertet und miteinander verglichen.

2 Problemstellung

2.1 Datenbeschreibung

Die vorliegenden Daten wurden im Zuge mehrerer Testfahrten durch das deutsche LTE-Netz der Netzbetreiber O2, T-Mobile und Vodafone im Raum Dortmund erhoben [5]. Die Testfahrten verliefen über vier zuvor festgelegte Routen, welche sich hinsichtlich der Art ihrer Umgebung unterscheiden:

- **Campus**: Direkte Umgebung der TU Dortmund, Routenlänge 3km.
- **Urban**: Stadtbereich, Routenlänge: 3km.

- **Suburban**: Vorstadtbereich, Routenlänge: 9km.
- **Highway**: Autobahn, Routenlänge: 14km.

Jede dieser Messfahrten wurde zehnmal wiederholt. Hierbei wurden sowohl passive Messungen der Netzqualität mithilfe verschiedener Indikatoren, als auch aktive Messungen der Up- und Downloadraten durchgeführt. Die Messungen der Datenübertragungsraten wurden alle 10s vollzogen, die Messungen der passiven Indikatoren alle 1s. Um die Datenübertragungsraten erfassen zu können, wurden Datenpakete zufälliger Größe von 0.1, 0.5, 1, ..., 10 MB an einen Server zur Messung übertragen. Die insgesamt erhobenen Variablen seien in der folgenden Auflistung kurz beschrieben:

- **RSRP:** *Reference Signal Received Power* gibt die Empfangsstärke eines Referenzsignals an. Je höher der Wert, desto besser ist der Empfang.
- **RSRQ:** *Reference Signal Received Quality* ist ein weiterer Indikator für die Verbindungsqualität. Er wird unter anderem aus dem RSRP berechnet und kann vom Funkmast verwendet werden, um die Notwendigkeit eines Funkmastwechsels abschätzen zu können.
- **SINR:** *Signal-to-interference-plus-noise Ratio* gibt das Verhältnis des tatsächlichen Signals zum Rauschen oder anderen Störeinflüssen an.
- **CQI:** *Channel Quality Indicator* ist ein Indikator, welcher Aufschluss über die Qualität des Übertragungskanals gibt.
- **TA:** *Timing Advance* gibt den Zeitversatz an, der zur Synchronisation zwischen Up- und Downlink verwendet wird. Damit gibt er indirekt Aufschluss über die Entfernung zum Funkmast.
- **f:** Gibt die *Frequenz* des LTE-Signals an.
- **Velocity:** Die Geschwindigkeit, mit der sich das Messgerät fortbewegt.
- **Cell ID:** Identifiziert eine Zelle im LTE-Netzwerk. Nicht zu verwechseln mit der eNodeB-ID, welche einen Funkmast identifiziert. Ein Funkmast kann mehrere Zellen haben.
- **Payload Size:** Die Größe des übertragenen Datenpakets zur Ermittlung der Datenübertragungsrate.
- **Data Rate:** Die gemessene Datenübertragungsrate. Es werden sowohl Upload- als auch Downloadraten gemessen.

Die Messungen aller Testfahrten lassen sich insgesamt zu vier verschiedenen Datensätzen zusammenfassen, welche auch später zur Bearbeitung der Projektziele verwendet werden:

- **Context**: Dieser Datensatz enthält die sekündlich durchgeführten passiven Messungen der Netzwerkindikatoren. Insgesamt enthält dieser Datensatz 68334 Messungen.
- **Cells**: Enthält Messungen des RSRP und RSRQ zu den Nachbarzellen der aktuell verbundenen Zelle. Dieser Datensatz enthält insgesamt 93443 Messungen.
- **Upload**: Dieser Datensatz enthält die Messungen der Upload-Raten, welche alle 10s durchgeführt werden zuzüglich der Indikatoren aus dem Context-Datensatz zum entsprechenden Zeitpunkt. Insgesamt enthält dieser Datensatz 6180 Messungen.
- **Download**: Analog zum Upload-Datensatz, nur dass hier die gemessenen Download-Raten erfasst wurden. Dieser Datensatz enthält insgesamt 6516 Messungen.

2.2 Zielsetzungen

2.2.1 Task I – Vorhersage der Datenübertragungsraten

In [5] wurde ein neuartiger Ansatz der datengetriebenen Simulation von Netzwerken (Data-driven Network Simulation, *DDNS*) vorgestellt, welcher darauf basiert, dass durch datengetriebene Modelle möglichst realitätsnahe Simulationen von Netzwerken erzeugt werden sollen. Ein Aspekt dieser Modelle besteht darin, dass Up- und Downloadraten abhängig von den übrigen Netzwerkindikatoren möglichst realistisch modelliert werden müssen. Hierzu werden Prädiktionsmodelle benötigt, welche diese Datenübertragungsraten entsprechend vorhersagen können.

Das erste Ziel dieses Projektes ist es nun, verschiedene Arten von Prädiktionsmodellen im Hinblick auf diese Problemstellung anzuwenden, und die Güte dieser Verfahren zu untersuchen. Hierbei wird auch analysiert, ob sich das Verhalten der Modelle bezüglich der verschiedenen Netzbetreiber und Testfahrtszenarien unterscheidet. Weiterhin wird auch die Relevanz der verwendeten Kovariablen untersucht.

2.2.2 Task II – Vorhersage der eNodeB-Verbindungsdauern

Bei den ersten Einsätzen von DDNS in [5] hat sich gezeigt, dass es oft zu großen Vorhersagefehlern kommt, wenn der Funkmast gewechselt wird (in der Fachsprache heißen LTE-Funkmasten auch *eNodeB*). Eine Idee, um diesem entgegenzuwirken ist, den Zeitpunk des eNodeB-Wechsels vorherzusagen. Kennt man diesen Zeitpunkt, könnte man diese Information im nächsten Schritt dazu verwenden, um die Prädiktionsmodelle zu verbessern.

Das zweite Ziel dieses Projektes ist also, die Restdauer der bestehenden Verbindung zu einer eNodeB und damit indirekt auch den Wechselzeitpunk zur nächsten eNodeB vorherzusagen. Auch hier wird die Güte des eingesetzten Prädiktionsmodells anschließend analysiert und das Verhalten des Modells bezüglich der verschiedenen Netzbetreiber und Einsatzszenarien, sowie die Relevanz der verwendeten Kovariablen untersucht.

3 Methodik

3.1 Allgemeine Vorgehensweise

3.1.1 Vorhersage der Datenübertragungsraten

Die Vorhersage der Datenübertragungsraten wird analog zu [5] ebenfalls mithilfe von Prädiktionsmodellen aus dem Bereich des Machine-Learning bzw. des statistischen Lernens vorgenommen. Hierbei kommen die Verfahren *Extreme Gradient Boosting* und *Lineare Regression mit ARMA-Fehlern* zum Einsatz, welche in den Abschnitten 3.2 und 3.3 näher beschrieben werden. Analog zu [5] wird auch hier für jeden Netzbetreiber ein eigenes Modell angepasst.

Der in Abschnitt 2.1 beschriebene Upload Datensatz, welcher sämtliche Kovariablen und die Zielvariable *Data Rate* enthält, dient hierbei als Basis zur Vorhersage der Upload-Datenraten. Analog dient der Download Datensatz zur Prädiktion der Download Datenraten. In beiden Fällen werden die Modelle also so angepasst, dass sie basierend auf den Kovariablen, also der gemessenen Netzwerkindikatoren, die Zielvariable *Data Rate* prognostizieren sollen.

3.1.2 Vorhersage der eNodeB-Verbindungsdauern

Zur Vorhersage der Verbindungsdauern zu einer eNodeB kommen neben den Messungen zur aktuellen LTE-Zelle auch Messungen zu den Nachbarzellen in Betracht. In diesem Fall liegen für die Nachbarzellen Messungen des RSRP sowie des RSRQ vor, welche sich im Cells Datensatz finden. Der Context Datensatz enthählt alle Messungen bezüglich der aktuell verbundenen Zelle und soll als Basis für die Prognose der eNodeB-Verbindungsdauern dienen. Da dieser Datensatz aber keine Informationen zu den Nachbarzellen enthält, müssen die Datensätze Context und Cells vor der Prädiktion noch zusammengeführt werden. Die eNodeB-IDs, welche zu diesem Zweck benötigt werden, lassen sich unmittelbar aus der *Cell ID* bestimmen. Das Zusammenführen der Datensätze geschieht so, dass zu jedem Zeitpunkt sowohl die Netzwerkindikatoren zur aktuell verbundenen eNodeB vorliegen, als auch die Informationen zum höchsten RSRP und RSRQ einer benachbarten eNodeB. Der Gedankengang hierbei ist, dass eine steigende Signalstärke bei einer benachbarten eNodeB möglicherweise Aufschluss über einen baldigen Verbindungswechsel geben könnte.

Die Zielvariable, also die Restdauer der aktuellen Verbindung zu einer eNodeB, kann leicht aus den Messdaten berechnet werden, indem man die Differenz des Zeitpunktes der aktuellen Messung zu dem Zeitpunkt der ersten zukünftigen Messung an einer neuen eNodeB bildet. Als Prädiktionsmodell der Verbindungsdauern kommt dann erneut das *Extreme Gradient Boosting* zum Einsatz, welches für jeden der drei Netzbetreiber separat angepasst wird.

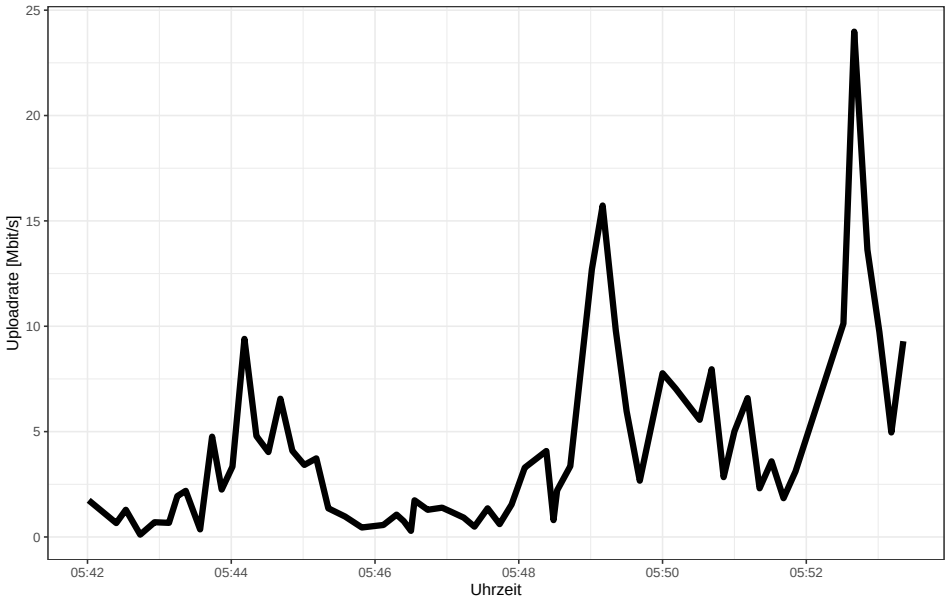

Abbildung 1: Die erste Messfahrt auf der Autobahn für den Netzbetreiber Vodafone am 12.12.2018.

3.1.3 Validierung und Tuning

Die Modellvalidierung erfüllt den Zweck, Aussagen darüber treffen zu können, wie sich die trainierten Modelle auf neuen und ungesehenen Daten, also beispielsweise zukünfig stattfindenden Messungen, verhalten werden. Ein bekanntes Verfahren dazu ist die k-fache Kreuzvalidierung [2], welche auch in [5] zum Einsatz gekommen ist. Hierbei wird der gesamte Datensatz zunächst zufällig in k gleich große Partitionen unterteilt, um im Anschluss das Modell jeweils auf $k-1$ Partitionen zu trainieren und die übrige Partition zum testen zu verwenden. Dies wird solange wiederholt, bis jede der k Partitionen genau einmal zum testen verwendet wurde. Obwohl dieses Verfahren sehr weit verbreitet ist, gibt es in der vorliegenden Situation jedoch Anhaltspunkte dafür, dass sich die k-fache Kreuzvalidierung möglicherweise als problematisch erweisen könnte.

In Abbildung 1 ist eine der durchgeführten Messfahrten einmal beispielhaft zu sehen. Man erkennt sofort, dass es sich bei den gemessenen Daten offenbar um eine Zeitreihe handelt. Würde man in dieser Situation eine k-fache Kreuzvalidierung einsetzen, bei der die Daten zufällig partitioniert werden, so würde der zeitliche Zusammenhang zwischen den Beobachtungen dadurch verloren gehen. Es wäre also fraglich, ob durch diese Art der Validierung verlässliche Aussagen über das Modellverhalten auf zukünftig erhobenen Messdaten getroffen werden können. Aus diesem Grund wurde in diesem Projekt ein eigenes Validierungsverfahren eingesetzt, welches speziell auf die vorliegende Situation zugeschnitten wurde.

In Abbildung 2 ist die in diesem Projekt eingesetzte Validierungsmethode einmal schematisch dargestellt. Wie bereits beschrieben, besteht der gesamte Datensatz an Messungen für einen Netzbetreiber aus zehn einzelnen Messfahrten für jedes der Szenarien *campus*, *highway*, *suburban* und *urban*. Jeder dieser Fahrten kann also chronologisch eine

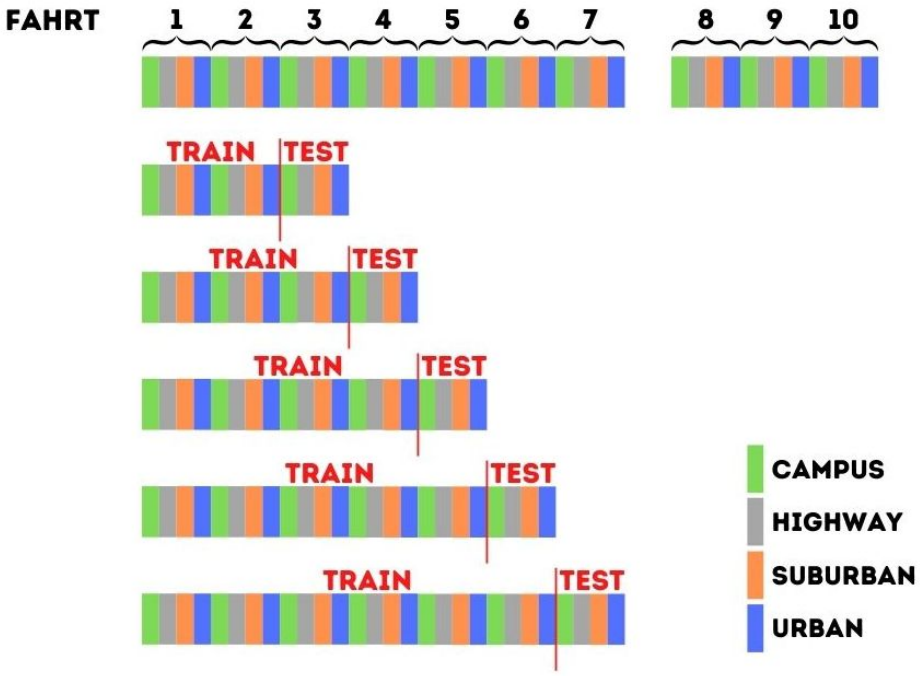

Abbildung 2: Das eingesetzte Verfahren zur Modellvalidierung.

Nummer von 1-10 zugewiesen werden, welche zusammen mit dem Szenario eine Fahrt eindeutig identifiziert. Im hier eingesetzten Validierungsverfahren wurde nun zunächst der gesamte Datensatz in zwei Teile aufgeteilt. Der erste Teil besteht aus den Fahrten 1-7, der zweite Teil besteht aus den Fahrten 8-10. In der Trainingsphase und beim Parametertuning kommt ausschließlich der erste Teil der Fahrten 1-7 zum Einsatz. So wird sichergestellt, dass das Modell beim Training keine Informationen aus zukünftigen Fahrten mit einbeziehen kann, wie es beispielsweise bei der k-fachen Kreuzvalidierung der Fall wäre. Fahrten 8-10 werden also ausschließlich zur Modellvalidierung eingesetzt.

Kennzahlen zur Evaluation der Vorhersagequalität Um die Vorhersagequalität der eingesetzten Verfahren zu bewerten, werden die Kennzahlen R^2, sowie der Mean Absolute Error (MAE) auf den Out-of-Sample Fahrten 8-10 wie folgt berechnet:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (t_i - r_i)^2}{\sum_{i=1}^{n} (t_i - \bar{t})^2} \tag{1}$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |t_i - r_i| \tag{2}$$

Hierbei ist n die Anzahl der vorhergesagten Datenpunkte, t_i gibt den wahren Wert der Zielgröße des i-ten Datenpunktes an, r_i gibt den vorhergesagten Wert für die Zielgröße des i-ten Datenpunktes an und $\bar{t}$ bezeichnet den Mittelwert der Zielgröße.

Tuning der Hyperparameter Das Tuning der Hyperparameter wird systematisch auf den Fahrten 1-7 mithilfe einer zufälligen Gittersuche vorgenommen. Dabei wird für jedes Modell ein fester Suchraum der Hyperparameter definiert, welcher entlang jeder Dimension in feste Gitterpunkte unterteilt wird. Das entstehende Gitter wird dann an einer festen Anzahl zufälliger Stellen ausgewertet und die zugehörigen Parameterkombinationen werden evaluiert.

Zur Evaluation einer Parameterkombination kommt, wie sich in Abbildung 2 ebenfalls erkennen lässt, eine Art Kreuzvalidierung für Zeitreihen zum Einsatz. Dabei wird der Trainingsdatensatz sukzessive um eine Fahrt erweitert und es wird immer auf der nächsten Fahrt getestet. Dies soll die Begebenheit simulieren, dass nach und nach neue Messfahrten vorgenommen werden, die den Gesamtdatensatz Schritt für Schritt erweitern.

Als Gütekriterium für eine getestete Parameterkombination wird der MAE verwendet. Die beste Kombination aus Hyperparametern wird dann im Anschluss auf dem gesamten Trainingsdatensatz der Fahrten 1-7 zur Modellanpassung genutzt und anschließend auf den Validierungsfahrten 8-10 evaluiert.

3.2 Extreme Gradient Boosting

Extreme Gradient Boosting ist ein Verfahren aus dem Bereich des maschinellen Lernens, welches sich in den letzten Jahren einer immer größeren Beliebtheit erfreut hat [1]. Eine sehr mächtige Implementierung, welche auch im Zuge dieses Projektes verwendet wurde, findet sich in der Open-Source Softwarebibliothek XGBoost[1]. Die grundlegende Funktionsweise dieses Verfahrens sei im Folgenden kurz beschrieben.

3.2.1 Ausgangssituation

Wir gehen davon aus, dass wir über einen Trainingsdatensatz $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ der Größe $|\mathcal{D}| = n$ verfügen, welcher aus den beobachteten Messungen $\mathbf{x}_i \in \mathbb{R}^m$ und der Zielgröße $y_i \in \mathbb{R}$ besteht, deren Wert wir vorhersagen wollen.

Das Ziel des Tree Boosting ist es, den Wert von y_i durch ein Ensemble von Entscheidungsbäumen (CART) vorherzusagen:

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F} \tag{3}$$

Hierbei ist $\mathcal{F}$ die Klasse der besagten Entscheidungsbäume, welche in jedem ihrer T Blätter einen konstanten Wert vorhersagen: $\mathcal{F} = \{f(\mathbf{x}) = w_{q(x)}\}$, wobei $q : \mathbb{R}^m \to T$ eine Funktion ist, die der Beobachtung $\mathbf{x}$ eines der T Blätter zuordnet und $w \in \mathbb{R}^T$ der Vektor der Blattvorhersagen (Gewichte) des Baumes ist.

[1] `https://github.com/dmlc/xgboost`

3.2.2 Zielfunktion

Die Zielfunktion, welche während des Trainings zur Anpassung des Modells minimiert wird, setzt sich wie folgt zusammen:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(\hat{y}_i, y_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{4}$$

Hierbei ist l eine differenzierbare und konvexe Verlustfunktion, welche Aufschluss über die Güte der Vorhersage $\hat{y}_i$ liefert. Ein Beispiel ist der quadratische Fehler, welcher durch $l(\hat{y}_i, y_i) = (\hat{y}_i - y_i)^2$ gegeben ist. Die Funktion Ω ist ein sogenannter Regularisierungs- oder Strafterm und ist wie folgt definiert:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{5}$$

Das Ziel von Ω ist es, eine zu hohe Komplexität der einzelnen Entscheidungsbäume in der Optimierung zu bestrafen und somit während des Trainings simplere Bäume zu bevorzugen. Dies geschieht mit dem Hintergedanken, eine Überanpassung des Modells an die Trainingsdaten verhindern zu wollen. Der Parameter γ bestraft hierbei die Anzahl der Blätter T eines Entscheidungsbaumes und der Parameter λ bestraft zu große Gewichte in den einzelnen Blättern.

3.2.3 Training

Das Grundprinzip des Boosting ist es, die Ensemble Modelle additiv nach dem Greedy-Prinzip zu trainieren. Dies funktioniert hier so, dass die einzelnen Entscheidungsbäume nicht alle gleichzeitig angepasst werden, sondern nach und nach zum Ensemble hinzugefügt werden. Jeder Baum, welcher in einem Schritt hinzugefügt wird, wird so trainiert, dass er die Zielfunktion soweit wie möglich minimiert.

Wenn im Optimierungsschritt t also der Entscheidungsbaum f_t zum Ensemble hinzugefügt wird, ergibt sich die folgende Verlustfunktion, welche durch f_t minimiert werden soll:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(\hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i), y_i) + \Omega(f_t) \tag{6}$$

Die Regularisierungsterme $\sum_{k=1}^{t-1} \Omega(f_k)$ der bereits zum Ensemble hinzugefügten Bäume wurden hierbei weggelassen, da sie im Zuge der Optimierung in Schritt t nicht mehr verändert werden können.

Beim Extreme Gradient Boosting wird $\mathcal{L}^{(t)}$ nun im Punkt $\hat{y}_i^{(t-1)}$ durch ein Taylor-Polynom 2. Grades approximiert, welches sich analytisch minimieren lässt. Streicht man alle konstanten Terme, welche für die Minimierung keine Rolle spielen, erhält man so die folgende Taylor-Approximation:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{7}$$

Hierbei sind $g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ und $h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ die erste und zweite partielle Arbleitung der Verlustfunktion l.

Wie in [1] gezeigt wurde, lassen sich dann die optimalen Gewichte $w_j^*, j = 1 \dots T$ für eine gegebene Baumstruktur q durch analytische Minimierung von $\tilde{\mathcal{L}}^{(t)}$ berechnen. Die Bestimmung einer optimalen Baumstruktur q hingegen ist rechnerisch durch Enumeration aller erdenklichen Möglichkeiten im Normalfall keine Option. Daher wird analog zum CART-Algorithms ein Greedy-Verfahren eingesetzt, welches den Baum durch sukzessives Hinzufügen neuer Verzweigungen aufbaut. Jede neue Verzweigung wird dabei so gewählt, dass der Wert von $\tilde{\mathcal{L}}^{(t)}$ durch die Bestimmung der optimalen Gewichte zum aktuellen Baum soweit wie möglich minimiert wird. Der Regularisierungsterm $\Omega(f_t)$ verhindert dabei direkt durch seine Anwesenheit in $\tilde{\mathcal{L}}^{(t)}$, dass die neue Baumstruktur zu komplex wird.

3.2.4 Hyperparameter

Die Hyperparameter des Extreme Gradient Boosting, welche im Rahmen dieses Projektes durch Tuning ermittelt wurden, seien im Folgenden kurz aufgelistet:

- n: Anzahl der Boosting-Runden und damit auch Anzahl der Bäume, die insgesamt zum Ensemble hinzugefügt werden. Je größer dieser Parameter gewählt wird, desto komplexer wird das Modell.
- γ: Regularisierungsterm, welcher die Anzahl an Blättern eines Entscheidungsbaumes bestraft. Dies soll die Bildung von zu komplexen Bäumen im Ensemble verhindern und somit eine Maßnahme gegen potenzielle Überanpassung sein.
- λ: Bestraft zu hohe Gewichte in den Blättern eines Entscheidungsbaumes, wird ebenfalls zur Vermeidung von Überanpassung verwendet.
- η: Wurde in einer Boosting-Runde ein neuer Baum gefunden, so geht er nur mit dem Gewicht η in das Gesamtensemble ein. Dies ist eine weitere Maßnahme, die gegen Überanpassung des Modells helfen soll.

Der Suchraum, welcher im Tuningprozess zur Ermittlung einer möglichst guten Kombination der Hyperparameter verwendet wurde, sei ebenfalls hier angegeben:

- $n \in [100, 1000]$
- $\gamma \in [0, 10]$
- $\lambda \in [0, 10]$
- $\eta \in [0.01, 1]$

Für die Gittersuche wurde dieser Suchraum entlang jeder Dimension in 20 Gitterpunkte unterteilt. Das resultierende Gitter wurde dann an 50 verschiedenen zufälligen Stellen ausgewertet.

3.2.5 Relevanz der Kovariablen

Um die Relevanz der einzelnen Kovariablen für ein Extreme Gradient Boosting Modell beurteilen zu können, wurde das Permutation Feature Importance Verfahren [4] verwendet. Bei diesem Verfahren wird zunächst ein beliebiges Gütemaß, z.B. der MAE, als Referenzwert für ein trainiertes Modell berechnet. Anschliessend wird eine der Kovariablen zufällig permutiert und es wird Änderung des Gütemaßes für den auf diese Weise manipulierten Datensatz untersucht. Die Idee dabei ist, eine Variable durch das zufällige Permutieren für das Modell unbrauchbar zu machen. Somit sollte sich das Gütemaß stark verschlechtern, wenn eine Variable permutiert wurde, die für das Modell sehr wichtig war. Bleibt das Gütemaß etwa gleich oder verändert sich nur geringfügig, so ist davon auszugehen, dass die Variable für das Modell nicht von besonders großer Relevanz war.

3.3 Lineare Regression mit ARMA-Fehlern

Die in diesem Projekt eingesetzte Variante der linearen Regression ist eine Erweiterung des klassischen linearen Modells, welche einen zeitlichen Zusammenhang zwischen den Fehlertermen als ARMA-Prozess abbilden kann. Eine vollständige Beschreibung dieses Verfahrens findet sich in [3], die Grundlagen seien hier aber im Folgenden kurz dargelegt.

3.3.1 Das lineare Modell

Das lineare Modell ist eins der bekanntesten Modelle aus dem Bereich des statistischen Lernens. Für einen Datensatz $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ der Größe $|\mathcal{D}| = n$ mit $\mathbf{x}_i \in \mathbb{R}^m$ und $y_i \in \mathbb{R}$ nimmt man bei dieser Methode an, dass zwischen y_i und x_i lediglich ein linearer Zusammenhang besteht:

$$y_i = \beta_0 + \sum_{j=1}^{m} \beta_j x_{ij} + \epsilon_i, \quad \epsilon_i \sim (0, \sigma^2) \tag{8}$$

Hierbei sind β_j die Modellkoeffizienten, welche aus den Daten geschätzt werden müssen, und ϵ_i sind die zufälligen Fehler des Modells. Wichtig ist es an dieser Stelle zu erwähnen, dass man bei der klassischen linearen Regression annimmt, dass die Fehler ϵ_i unabhängig voneinander verteilt sind und alle den Mittelwert 0 sowie identische Varianz σ^2 haben.

Da es sich in der vorliegenden Situation jedoch um Zeitreihen handelt, kann nicht davon ausgegangen werden, dass die Annahme der Unabhängigkeit erfüllt ist. Aus diesem Grund wurde in diesem Projekt eine Erweiterung der linearen Regression eingesetzt, welche es zulässt, dass die Fehlerterme nicht notwendigerweise unabhängig sein müssen. Dies wird dadurch erreicht, dass man die Fehlerterme ϵ_i im linearen Modell durch ein sogenanntes *Autoregressives Moving-Average (ARMA)* Modell abbildet, welches einen zeitlichen Zusammenhang zwischen den Fehlern modellieren kann.

3.3.2 Erweiterung durch ARMA-Prozesse

Die Modellgleichung des ARMA-Prozesses einer Zeitreihe $(\epsilon_i)_{i=1}^n$ ist gegeben durch:

$$\epsilon_i = \underbrace{\sum_{j=1}^{p} \phi_j \epsilon_{i-j}}_{\text{AR-Teil}} + \underbrace{\sum_{j=1}^{q} \theta_j \eta_{i-j}}_{\text{MA-Teil}} + \eta_i, \quad \eta_i \sim (0, \sigma^2) \tag{9}$$

Wie in der Gleichung 9 schon dargestellt, setzt sich dieser Prozess aus zwei Komponenten zusammen. Der Teil $\sum_{j=1}^{p} \phi_j \epsilon_{i-j}$ bezeichnet hierbei die Autoregressive Komponente des ARMA-Prozesses der Ordnung p, der Teil $\sum_{j=1}^{q} \theta_j \eta_{i-j}$ die Moving-Average Komponente der Ordnung q. Weiterhin bezeichnet η_i den jeweiligen Modellfehler zum Zeitpunkt i, wobei die Modellfehler als unabhängig und identisch verteilte Zufallsvariablen mit Mittelwert 0 und Varianz σ^2 angenommen werden.

Bei der linearen Regression mit ARMA-Fehlern nimmt man nun an, dass die Vorhersagefehler der Regression, also die ϵ_i, einem solchen ARMA-Prozess gehorchen. Erweitert man die Modellgleichung der linearen Regression entsprechend, so erhält man die Gesamtgleichung der linearen Regression mit ARMA-Fehlern:

$$y_i = \beta_0 + \sum_{j=1}^{m} \beta_j x_{ij} + \sum_{j=1}^{p} \phi_j \epsilon_{i-j} + \sum_{j=1}^{q} \theta_j \eta_{i-j} + \eta_i, \quad \eta_i \sim (0, \sigma^2) \tag{10}$$

Die Ordnungsparameter p und q sind hierbei die Hyperparameter des Modells und müssen vom Anwender sinnvoll gewählt werden. In diesem Projekt wurde hierzu wie in Abschnitt 3.1.3 schon im Detail beschrieben, eine Gittersuche als Tuningalgorithmus eingesetzt.

3.3.3 Parameterschätzung und Modellannahmen

Zur Schätzung der Modellparameter β_i, ϕ_j und θ_j gibt es im Allgemeinen mehrere Möglichkeiten [3]. Das in diesem Projekt verwendete Softwarepaket *forecast*[2] setzt hierzu die Maximum-Likelihood Methode ein. Bei dieser Methode wird unter Annahme einer konkreten Verteilung der η_i die Wahrscheinlichkeit maximiert, mit welcher die jeweils Vorliegende Stichprobe aufgetreten ist. Die Maximierung dieser Wahrscheinlichkeit wird dann durch Optimierung der Modellparameter durchgeführt.

Die gängigste Verteilungsannahme, welche auch im *forecast* Paket implementiert ist, ist die Normalverteilungsannahme. Hierbei geht man konkret davon aus, dass $\eta_i \sim \mathcal{N}(0, \sigma^2)$ gilt. Um aussagekräftige Ergebnisse zu erhalten, muss diese Annahme allerdings vorher überprüft werden. Hierzu eignen sich beispielsweise sogenannte qq-Plots, welche die Quantile der beobachteten Verteilung gegen die theoretischen Quantile einer Normalverteilung abtragen. Verhält sich der Zusammenhang zwischen den gemessenen und den theoretischen Quantilen in etwa wie eine Gerade, so kann man davon ausgehen, dass die Normalverteilungsannahme erfüllt ist.

[2] `https://pkg.robjhyndman.com/forecast/`

Zur Überprüfung, ob dies auch in der vorliegenden Situation gegeben ist, ist in Abbildung 3 für jeden der betrachteten Netzbetreiber ein QQ-Plot der Residuen aufgeführt. Man erkennt, dass grundsätzlich ein linearer Zusammenhang zwischen den theoretischen und den empirischen Quantilen vorhanden zu sein scheint. Es lässt sich zwar ebenfalls erkennen, dass es mitunter an den Rändern leichte Abweichungen von der Gerade gibt, jedoch wird der lineare Zusammenhang dadurch nicht übermäßig verletzt. Da man in praktischen Anwendungen sowieso nur selten auf perfekt normalverteilte Daten trifft, wird an dieser Stelle unter der Prämisse weitergearbeitet, dass die Normalverteilungsannahme als hinreichend erfüllt angesehen werden kann.

4 Ergebnisse

4.1 Vorhersage der Datenübertragungsraten

4.1.1 Extreme Gradient Boosting

Die Out-of-Sample Vorhersagen der Datenübertragungsraten des Extreme Gradient Boosting Modells finden sich in Abbildung 4. Man erkennt, dass sich die Verteilungen der Datenraten mitunter stark je nach Szenario und Anbieter unterscheiden. Vor allem fällt auf, dass der Anbieter Vodafone eine wesentlich höhere Variation in den Download-Raten vorweist, alls die übrigen Anbieter. Insgesamt lassen sich anhand dieser Vorhersagen allerdings keine systematischen Unregelmäßigkeiten erkennen.

4.1.2 Regression mit ARMA-Fehlern

Für die lineare Regression mit ARMA-Fehlern finden sich die Out-of-Sample Vorhersagen in Abbildung 5. Vergleicht man diese mit den Vorhersagen des Extreme Gradient Boosting, so fallen hier schon etwas stärkere systematische Abweichungen ins Auge. Beispielsweise scheint es so, dass im *urban* Szenario für den Anbieter O2 höhere Upload-Raten systematisch unterschätzt und niedrigere Upload-Raten systematisch überschätzt werden. Dies gibt einen ersten Aufschluss darüber, dass das Modell möglicherweise nicht expressiv genug sein könnte, um die Zusammenhänge in den Daten zu erfassen.

4.1.3 Modellvergleich

Die betrachteten Kennzahlen R^2 und MAE wurden in Abbildung 6 einander gegenübergestellt. Man erkennt sofort, dass Extreme Gradient Boosting für jeden Anbieter die besseren Werte liefert, als die lineare Regression mit ARMA-Fehlern. Dies würde die Vermutung bestätigen, dass das Modell der ARMA-Regression nicht in der Lage ist, sämtliche Zusammenhänge in den Daten zu erfassen und das somit das Extreme Gradient Boosting besser zur Vorhersage der Datenübertragungsraten geeignet ist.

Die Relevanz der einzelnen Kovariablen für die beiden Prädiktionsmodelle wurde in Abbildung 7 dargestellt. Diese wurden bei der linearen Regression mit ARMA-Fehlern durch die normierte absolute Größe der Modellkoeffizienten ermittelt. Beim Extreme

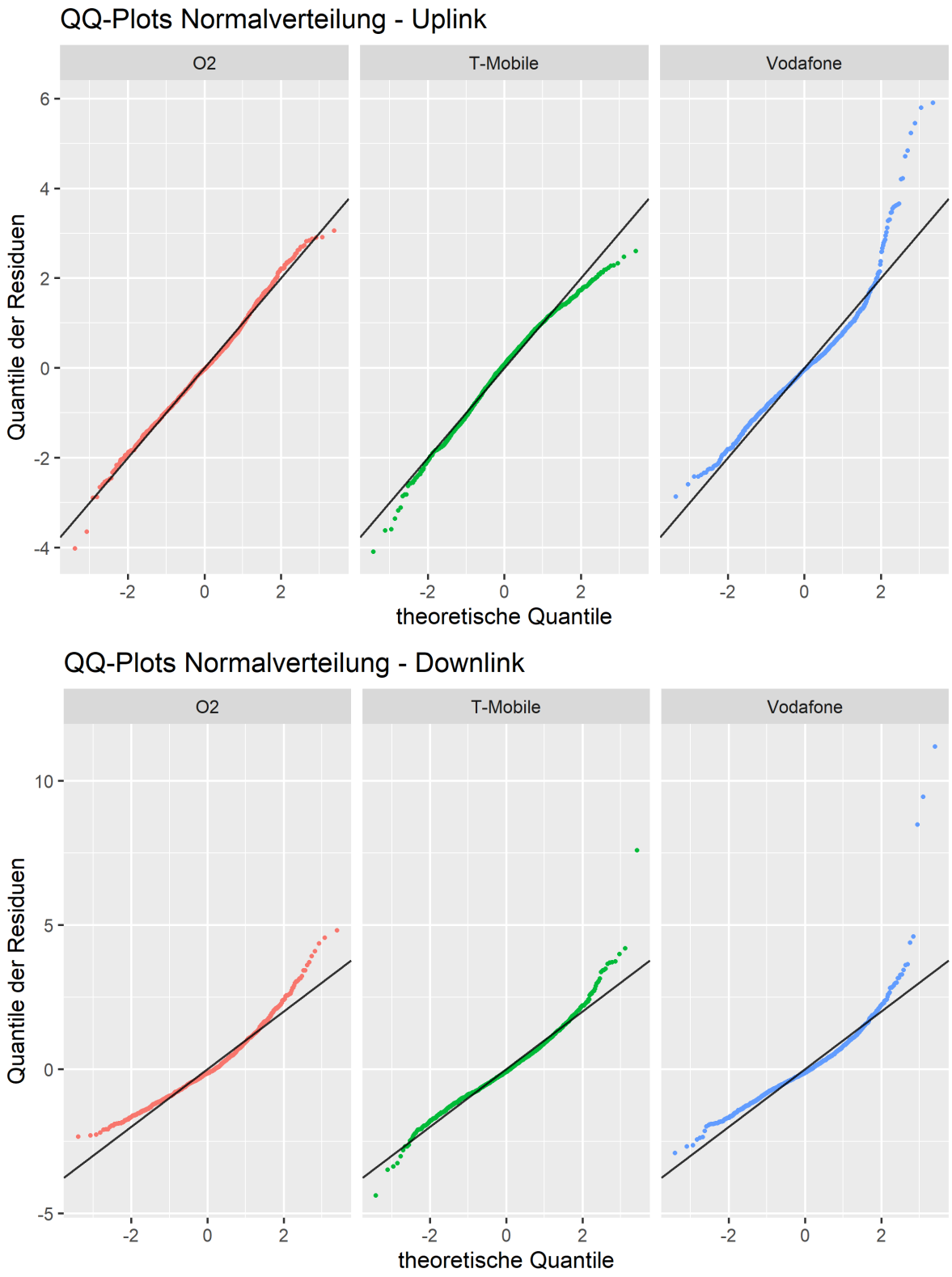

Abbildung 3: QQ-Plots der Residuen für die verschiedenen Netzbetreiber.

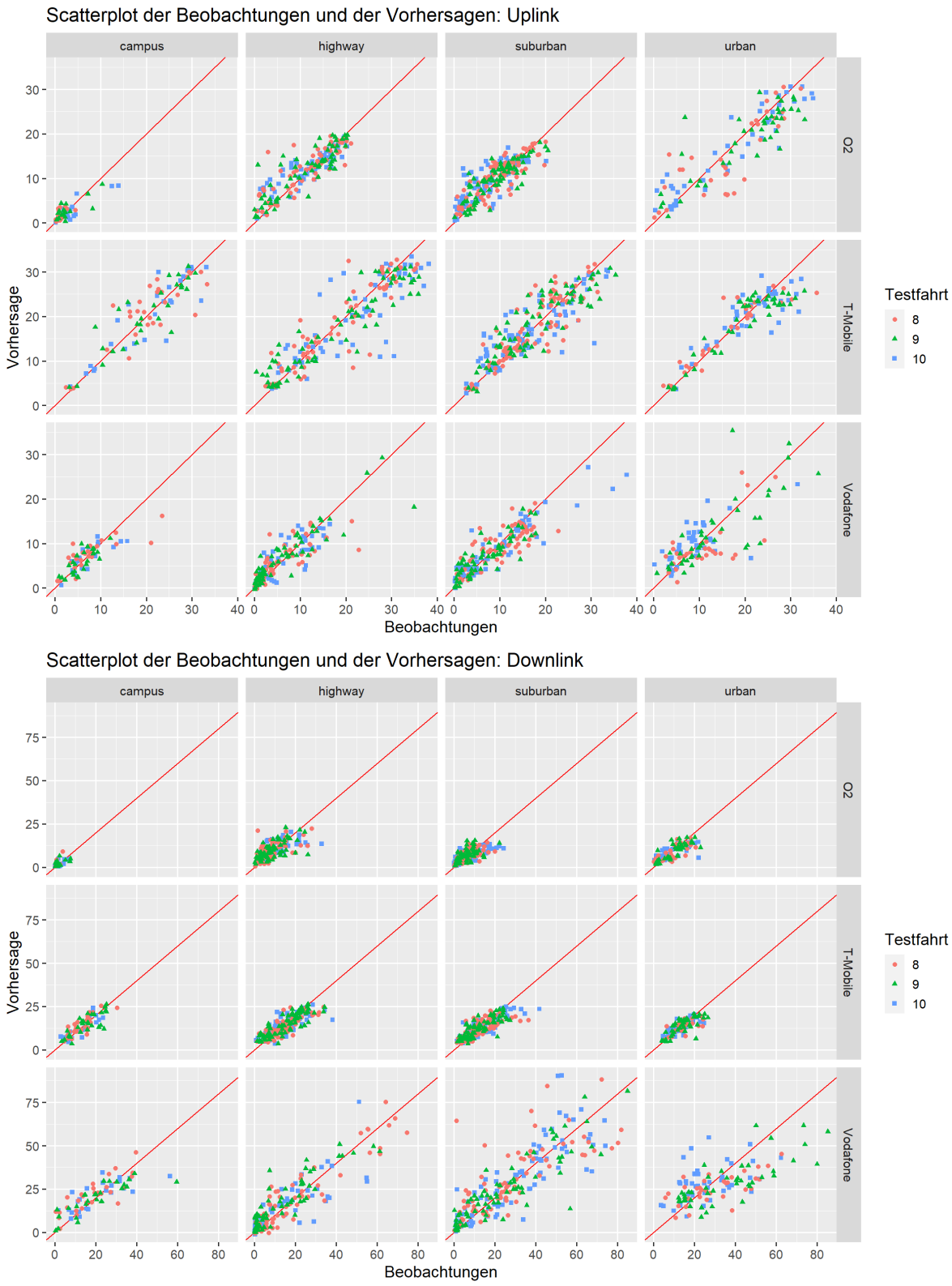

Abbildung 4: Out-of-Sample Vorhersagen der Datenraten für Extreme Gradient Boosting.

Abbildung 5: Out-of-Sample Vorhersagen der Datenraten für die Regression mit ARMA-Fehlern.

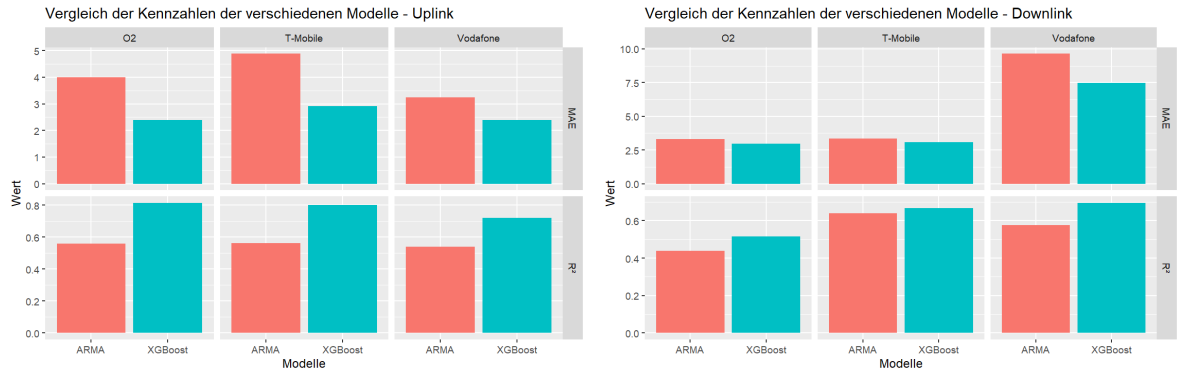

Abbildung 6: Vergleich der Kennzahlen für die Prädiktion der Upload- und Download-Raten.

Gradient Boosting kam die Permutationsmethode [4] zum Einsatz, die Ergebnisse daraus wurden zur besseren Vergleichbarkeit ebenfalls normiert.

Hierbei fällt auf, dass der Variable *Payload* in jeder Situation eine hohe Relevanz besitzt. Für Externe Gradient Boosting wird dies sogar besonders deutlich, hier erhält *Payload* in jeder Situation den höchsten Wichtigkeitswert.

4.2 Vorhersage der eNodeB-Verbindungsdauern

Zur Vorhersage der eNodeB-Verbindungsdauern wurde ausschließlich das Extreme Gradient Boosting Modell eingesetzt. Die Out-of-Sample Vorhersagen hierzu finden sich in Abbildung 8.

5 Zusammenfassung

Dies und das...

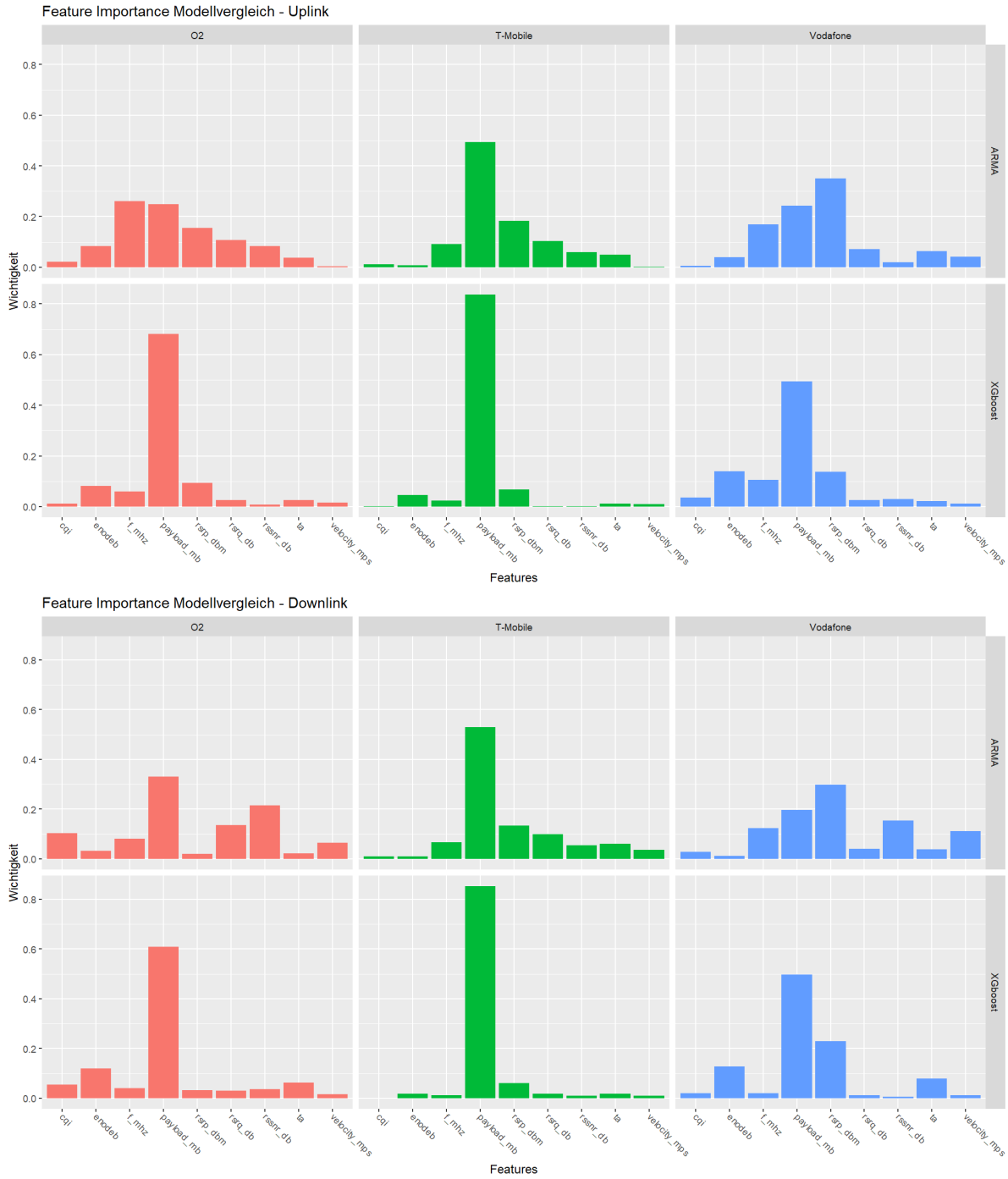

Abbildung 7: Relevanz der Kovariablen bezüglich der beiden Modelle zur Datenratenprädiktion.

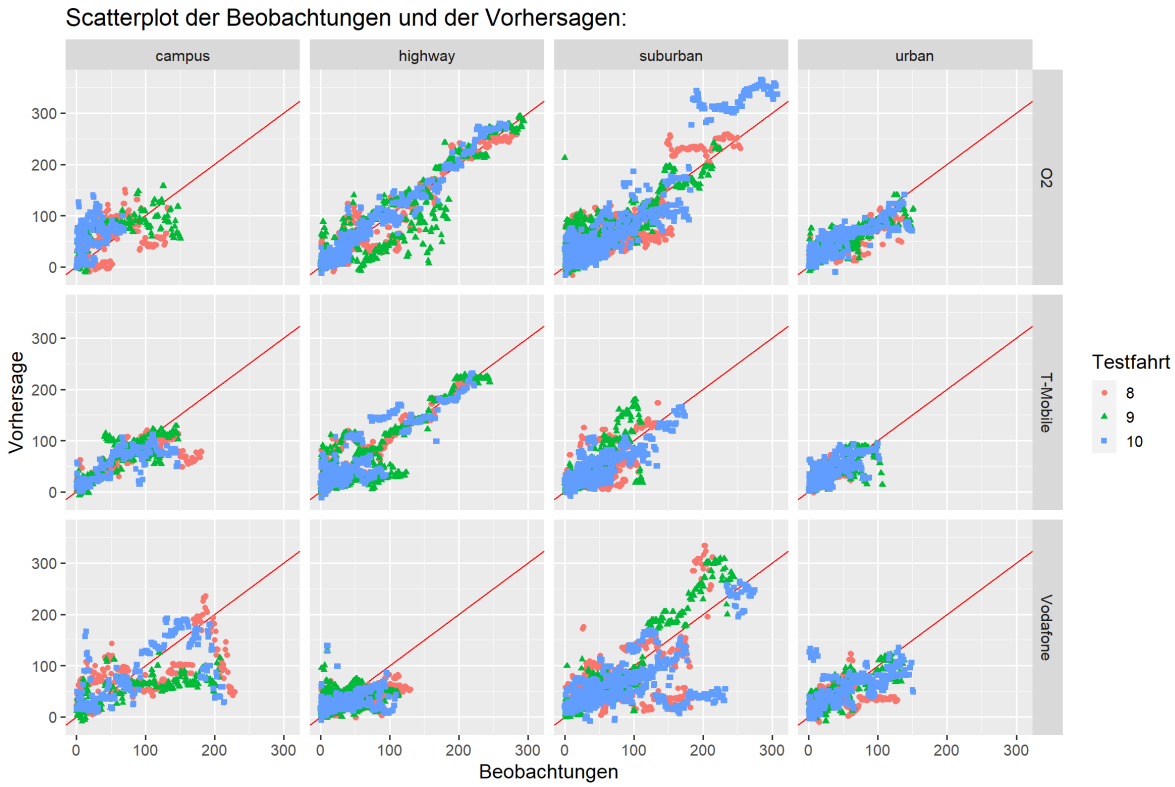

Abbildung 8: Out-of-Sample Vorhersagen der eNodeB-Verbindungsdauern.

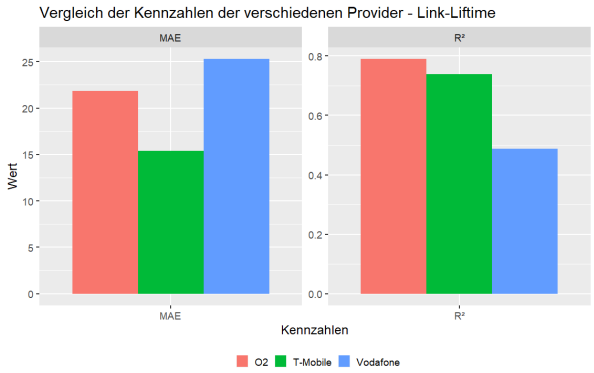

Abbildung 9: Kennzahlen Link-Lifetime.

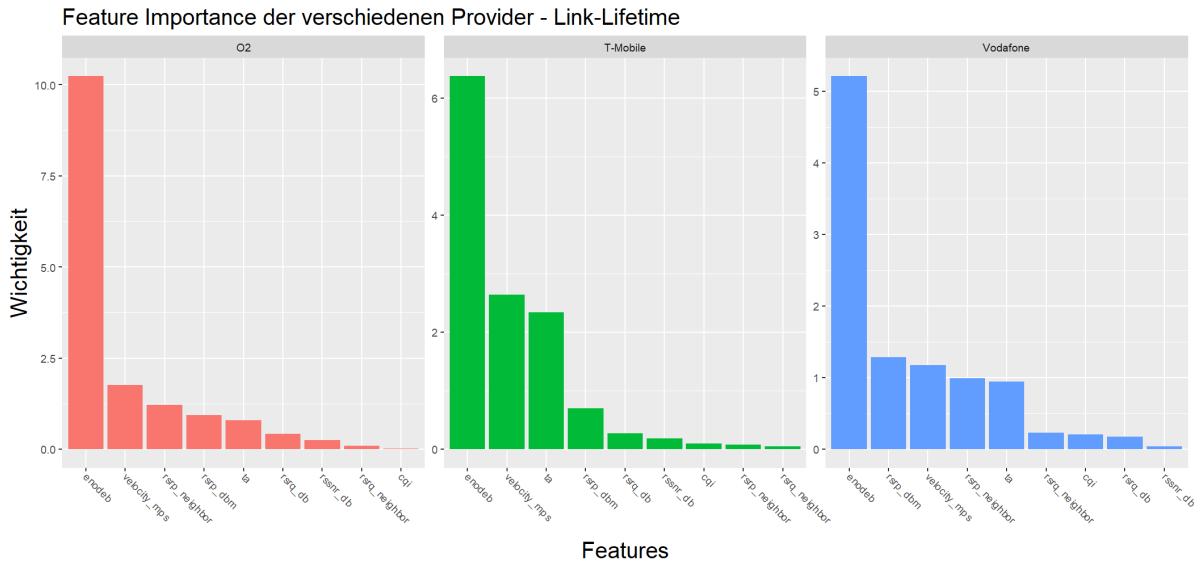

Abbildung 10: Feature Importance Link-Lifetime.

Literatur

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[3] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts: Melbourne, Australia, 2 edition, 2018. `https://otexts.com/fpp2/`.

[4] Christoph Molnar. *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[5] B. Sliwa and C. Wietfeld. Data-driven network simulation for performance analysis of anticipatory vehicular communication systems. *IEEE Access*, 7:172638–172653, 2019.