Fallstudien II

Laura Kampmann, Christian Peters, Alina Stammen

18. Dezember 2020

Inhalt

1. Task I - Vorhersage der Datenrate
    - Extreme Gradient Boosting
    - Regression mit ARMA-Fehlern
    - Modellvergleich

2. Task II - Handover Vorhersage und Link Lifetime
    - Feature Importance

3. Ausblick

Task I - Vorhersage der Datenrate

Extreme Gradient Boosting

Extreme Gradient Boosting

- Additives Training eines Ensembles aus „schwachen “ Lernern
  - $\Rightarrow$ In unserem Fall einfache CART-Bäume
- Jeder neue Baum versucht, die Schwächen seiner Vorgänger auszugleichen
  - $\Rightarrow$ Mit jedem neuen Baum sinkt der Training-Error
- Implementiert in *XGBoost* Bibliothek
  - Sehr gut skalierbar, funktioniert noch problemlos mit mehreren Milliarden Samples
  - Lässt sich aber auch hervorragend auf ressourcenbegrenzten Systemen einsetzen [1]

Features

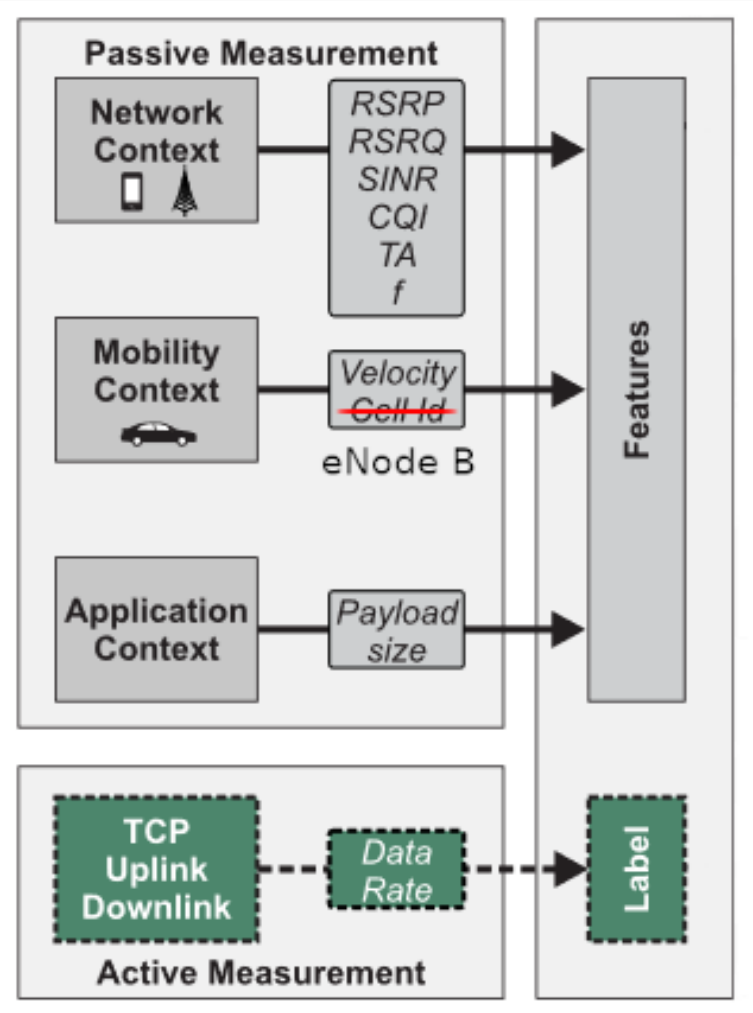


Abbildung 1: Modellfeatures [4].

Validierung

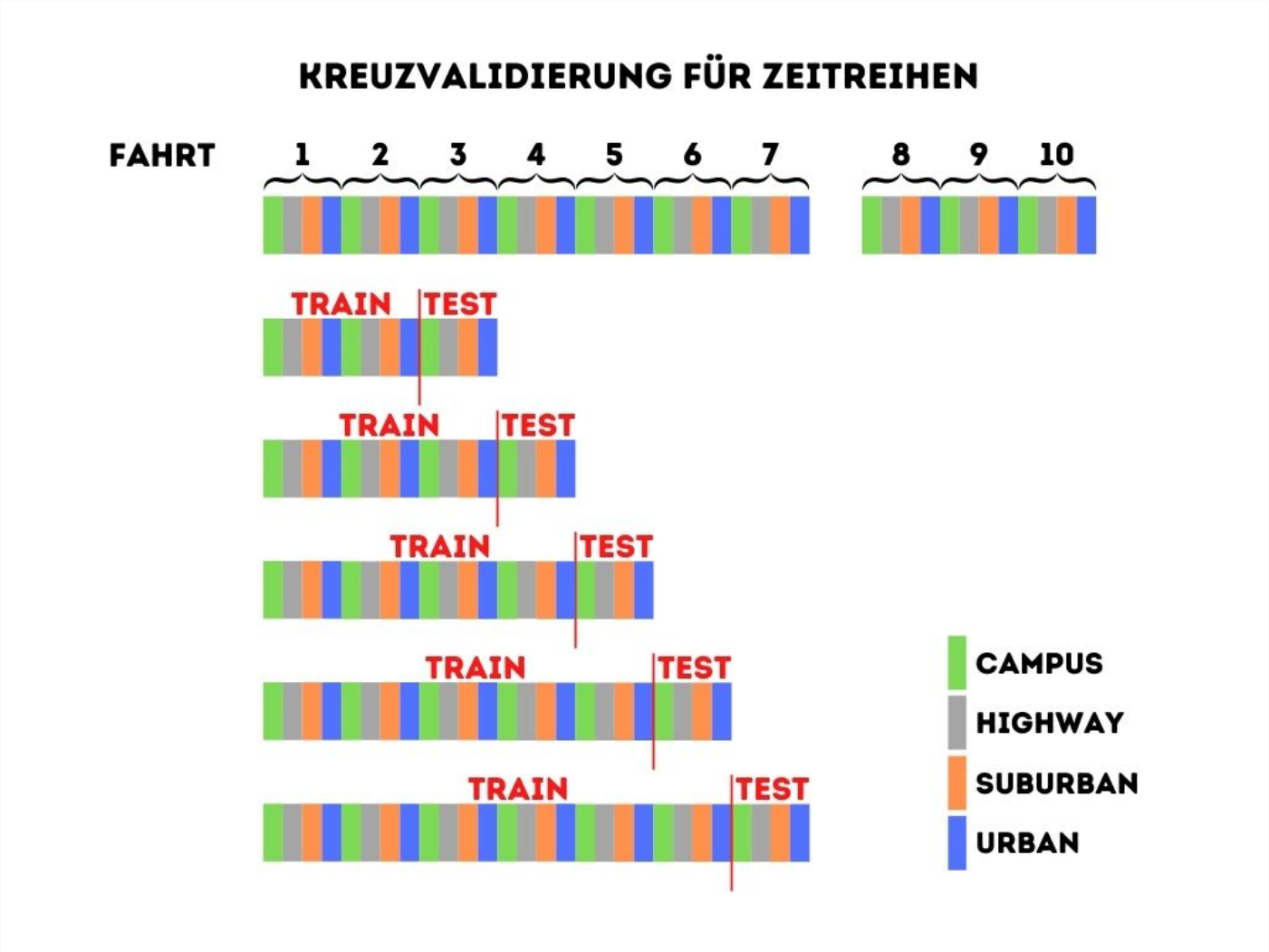


Abbildung 2: Einteilungen in Trainings- und Testdatensätze bei der Kreuzvalidierung für Zeitreihen.

Tuning

Suchraum der Hyperparameter:

- Anzahl der Boosting Runden $n\_rounds \in [100, 1000]$
- „Shrinkage “ Faktor (Lernrate) $\eta \in [0.01, 1]$
- Strafterm für Anzahl Baumblätter $\gamma \in [0, 10]$
- Strafterm für Vorhersagen der Baumblätter $\lambda \in [0, 10]$

$\Rightarrow$ **Randomisierte Gittersuche**

- 20 Gitterpunkte in jeder Dimension
  $\Rightarrow$ Insgesamt $20^4 = 160.000$ Gitterpunkte
- Ausgewertet an 50 zufälligen Stellen
- Berechnung des MAE mit Zeitreihenkreuzvalidierung für die Fahrten 1-7

Out-of-Sample Vorhersagen Upload

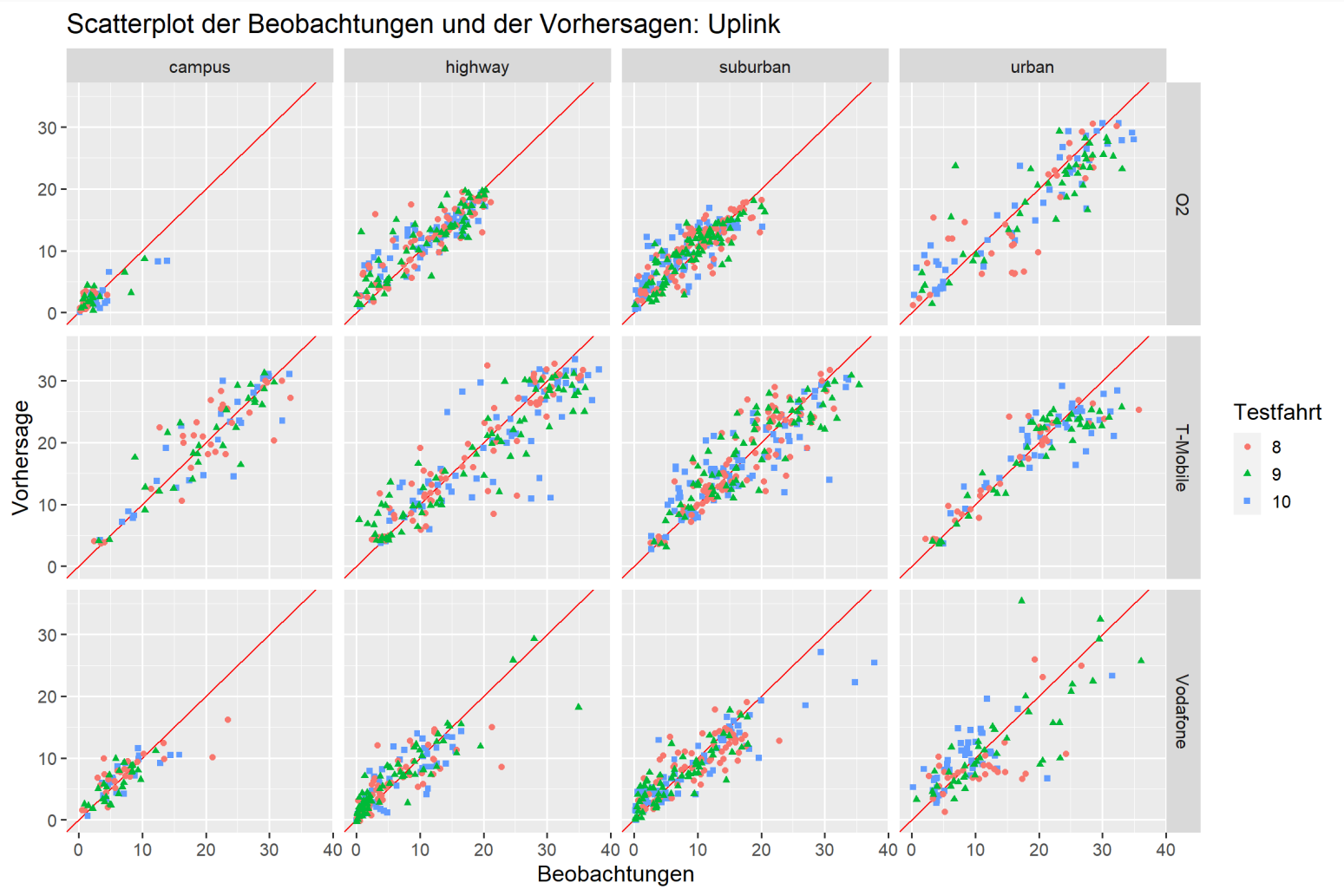


Abbildung 3: XGBoost Out-of-Sample Vorhersagen der Upload-Rate

Out-of-Sample Vorhersagen Download

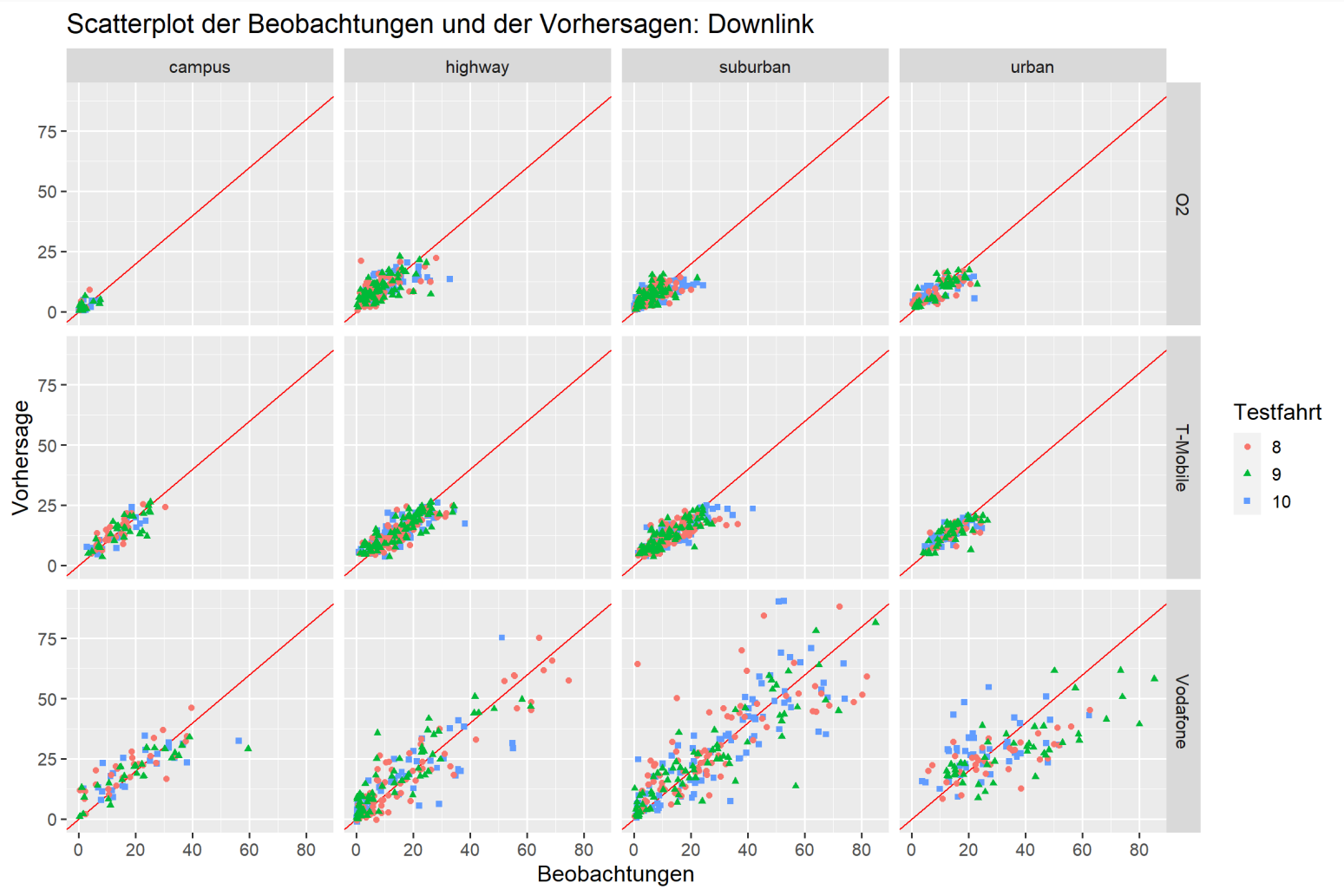


Abbildung 4: XGBoost Out-of-Sample Vorhersagen der Download-Rate

Task I - Vorhersage der Datenrate

Regression mit ARMA-Fehlern

Regression mit ARMA-Fehlern

Gegeben:

- Beobachtungen $(y_1, ..., y_T)$ der Zeitreihe $(y_t)_t$
- Beobachtungen $(x_1^{(i)}, ..., x_T^{(i)})$ der Zeitreihen $(x_t^{(i)})_t$ für $i = 1, ..., k$

Modellgleichung: Regression mit ARMA(p, q)-Fehlern [3]

$$y_t = c + \sum_{j=1}^{k} \beta_j x_t^{(j)} + \eta_t \text{ mit}$$

$$\eta_t = \underbrace{\sum_{k=1}^{p} \phi_p \eta_{t-p}}_{\text{vergangene Fehler: LM}} + \underbrace{\sum_{l=1}^{q} \theta_l \epsilon_{t-q}}_{\text{vergangene Fehler: ARMA}} + \epsilon_t$$

Regression mit ARMA-Fehlern

Vorarbeit:

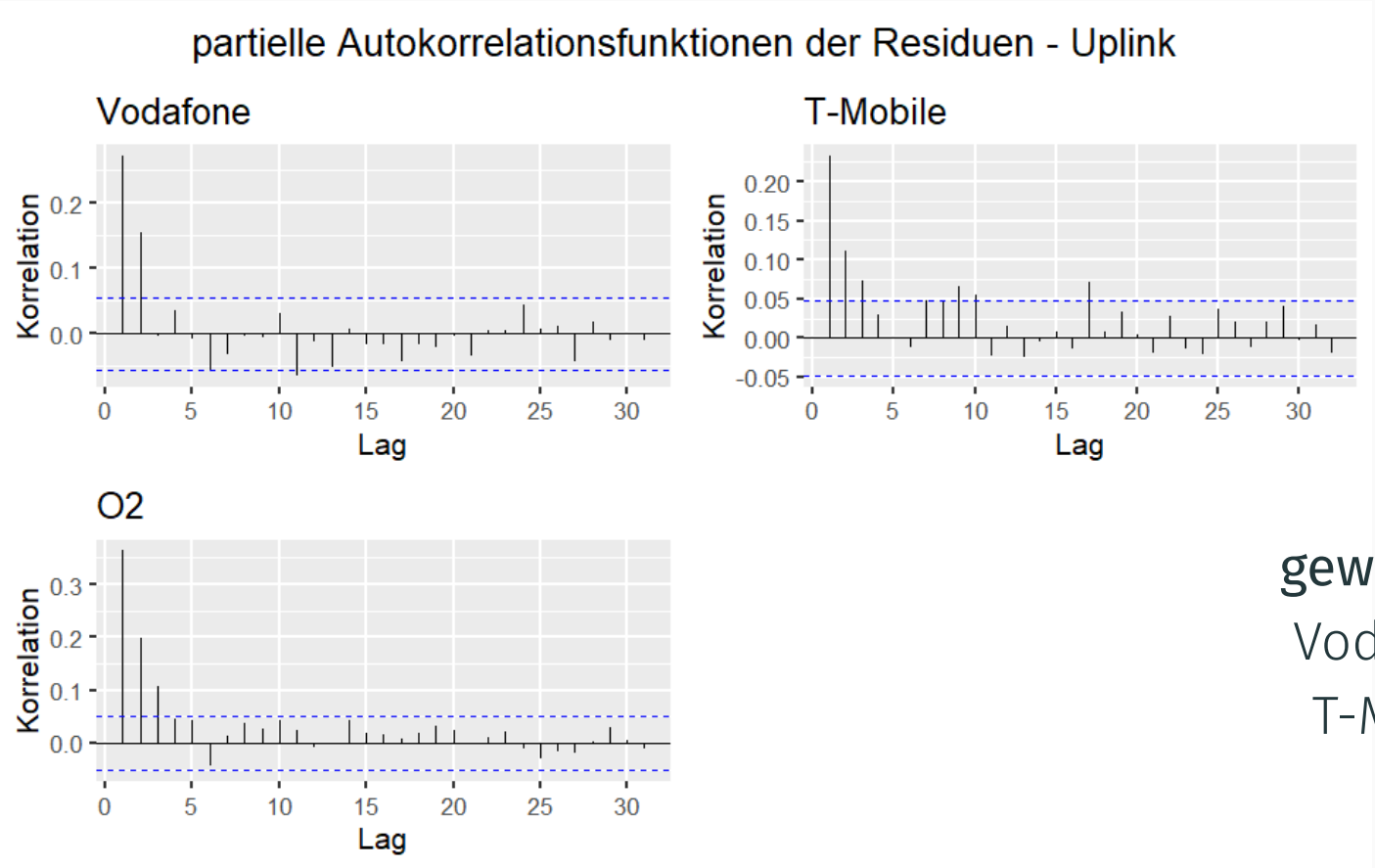
- Überprüfung Autokorrelation der Zielvariablen (Acf, pAcf)
- Standardisierung Train, Skalierung Test

Überprüfung der Voraussetzungen:

- Stationarität aller Variablen (Augmented Dickey-Fuller Test)
- keine Multikollinearität vorhanden (VIF)
- Normalverteilung der Residuen (Scatterplot, Histogramm, QQ-Plot)

Regression mit ARMA-Fehlern

Bestimmung des Grids für die AR-Ordnung - Uplink



gewählte Grids:

Vodafone: 0 - 2

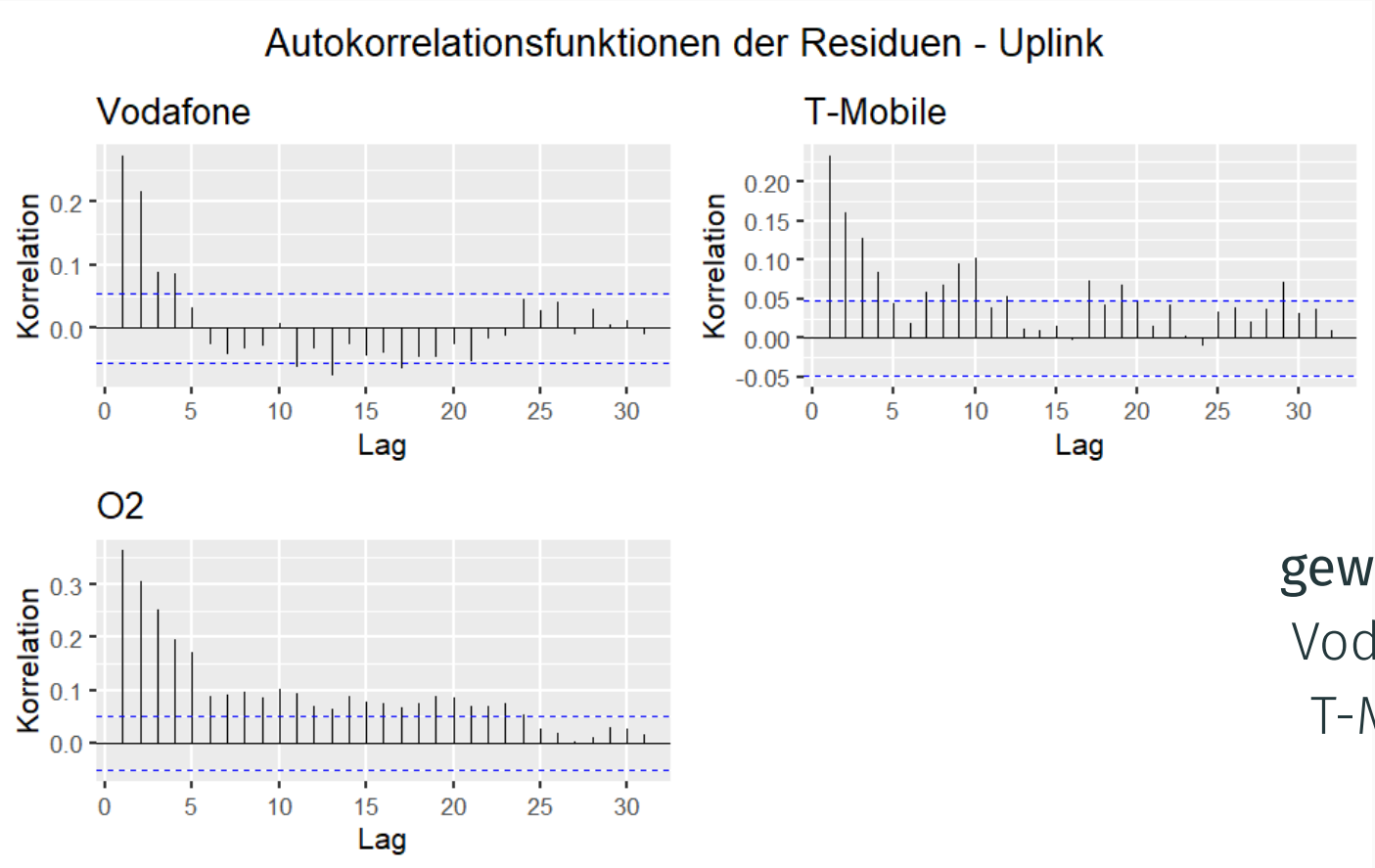
T-Mobile: 0 - 3

O2: 0 - 3

Abbildung 5: Partielle Autokorrelationsfunktion der Residuen des linearen Modells in Richtung Uplink.

Regression mit ARMA-Fehlern

Bestimmung des Grids für die MA-Ordnung - Uplink



gewählte Grids:

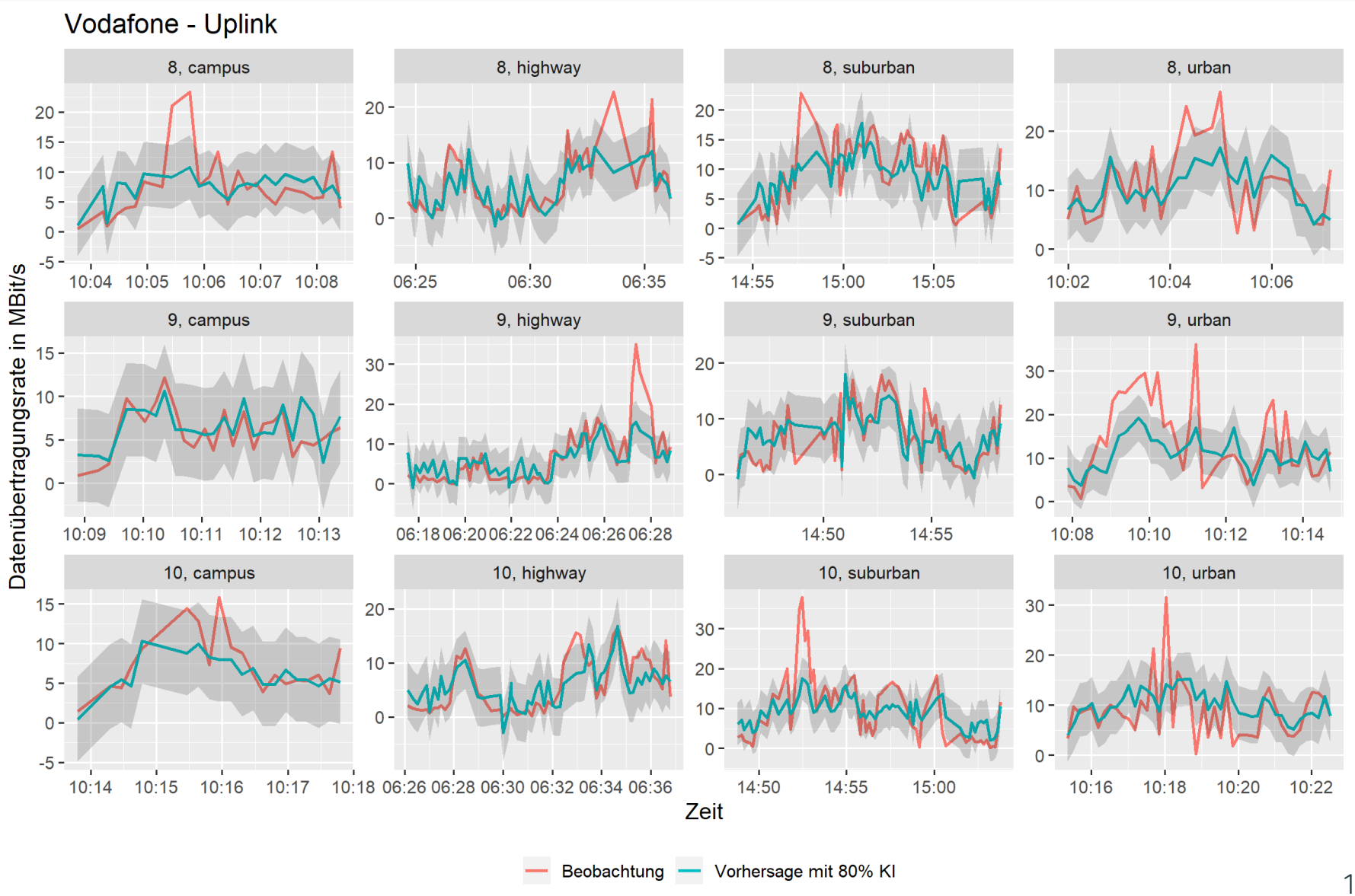
Vodafone: 0 - 4

T-Mobile: 0 - 4

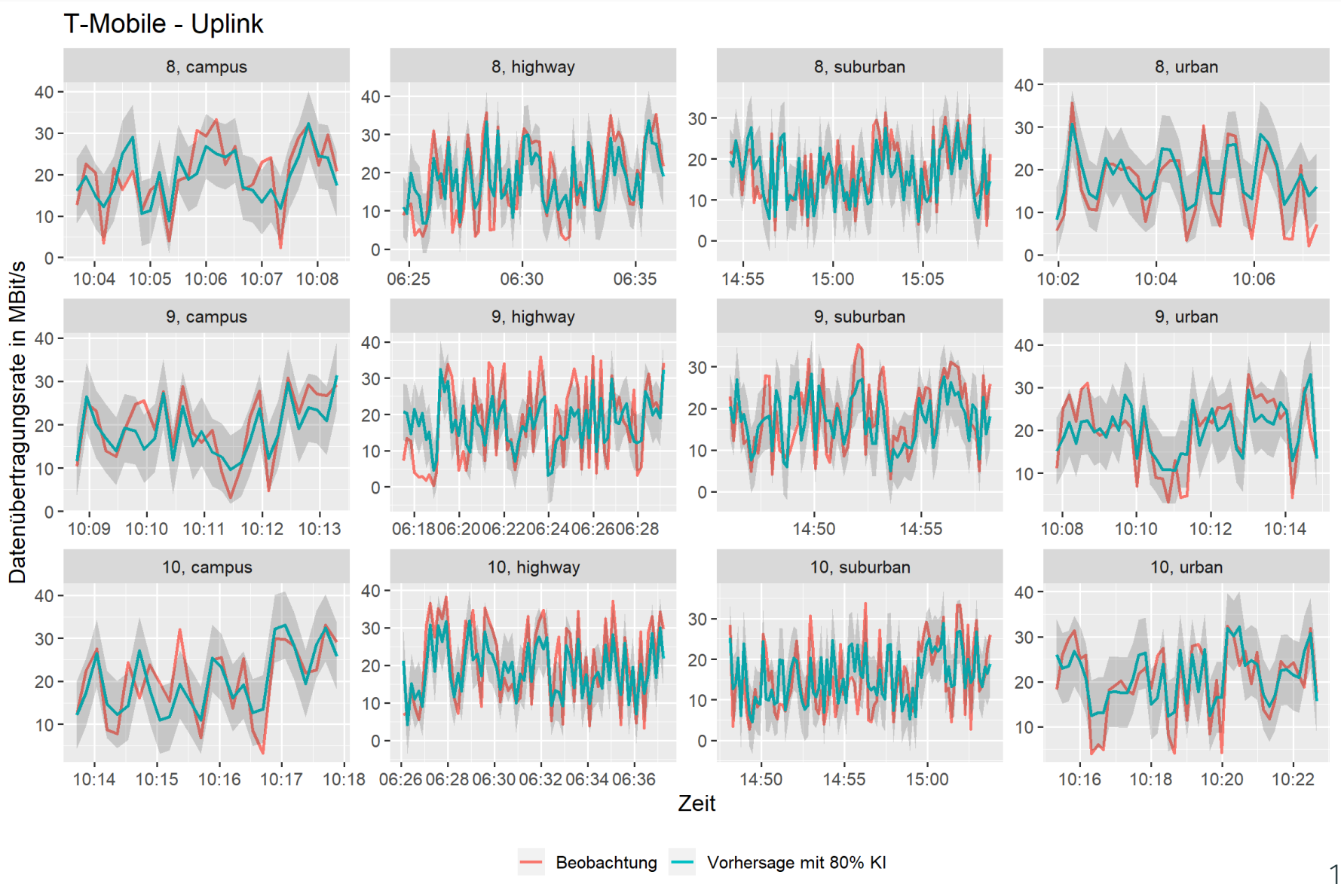
O2: 0 - 5

Abbildung 6: Autokorrelationsfunktion der Residuen des linearen Modells in Richtung Uplink.

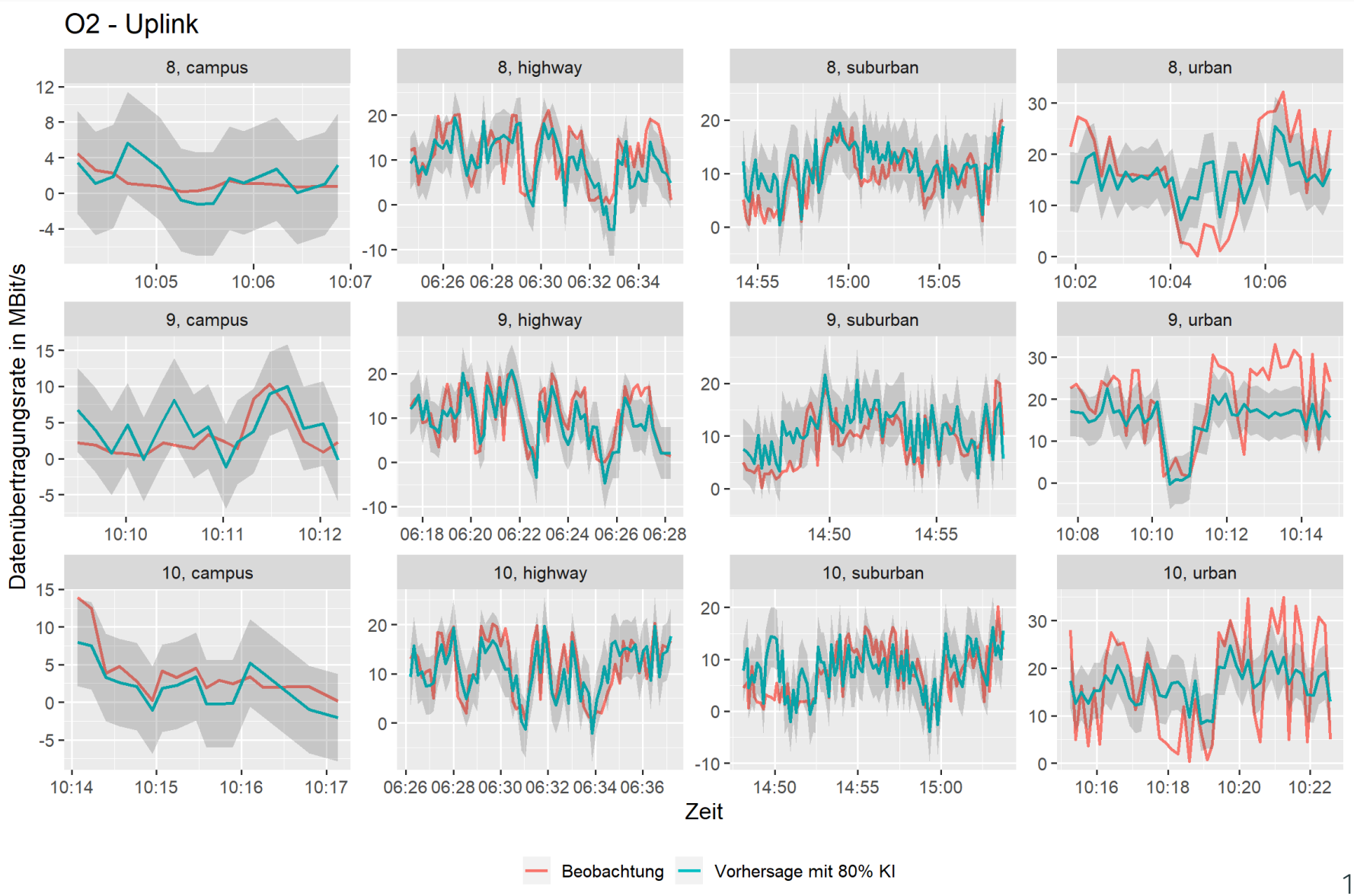
Regression mit ARMA-Fehlern: Ergebnisse (Uplink)



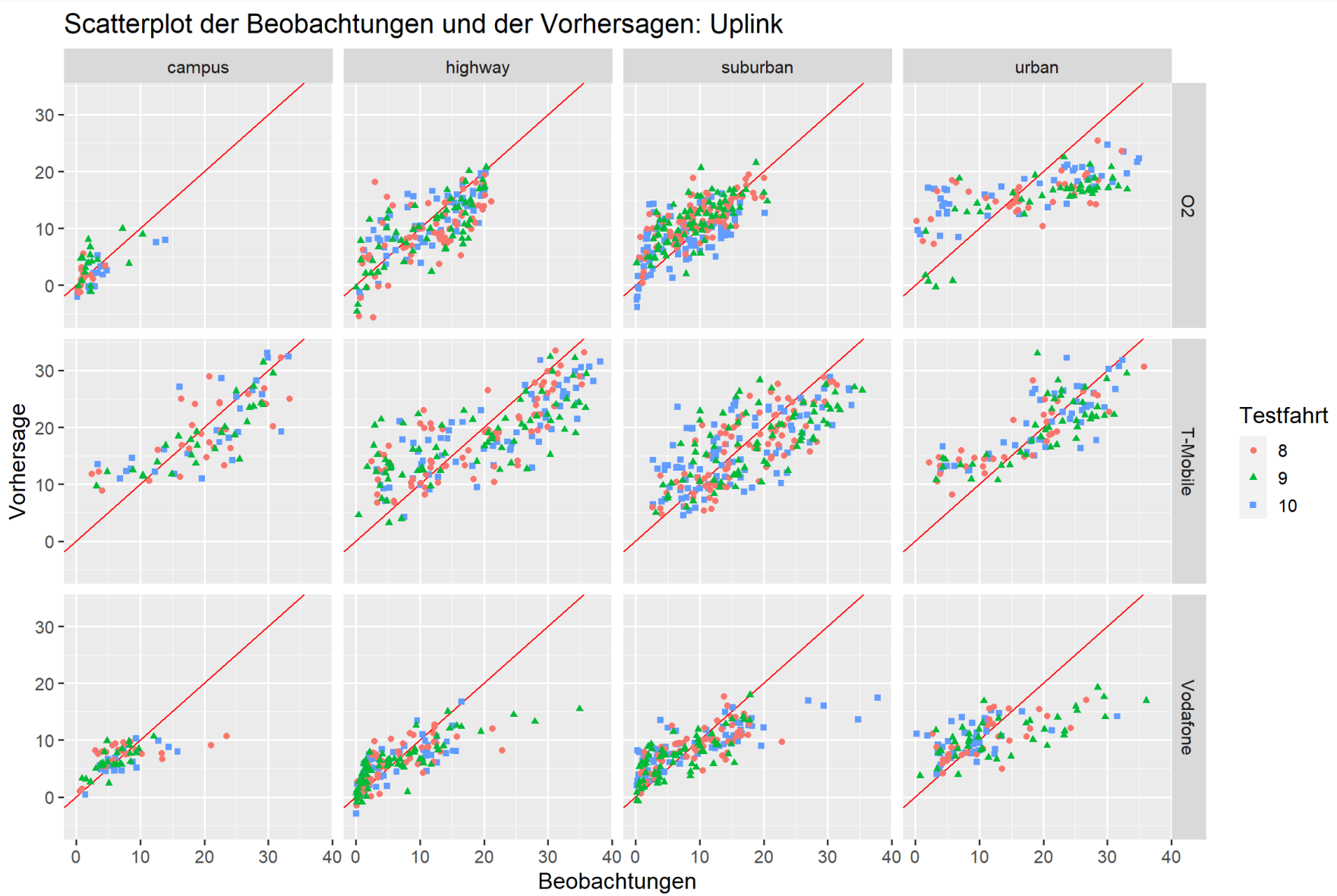
Regression mit ARMA-Fehlern: Ergebnisse (Uplink)



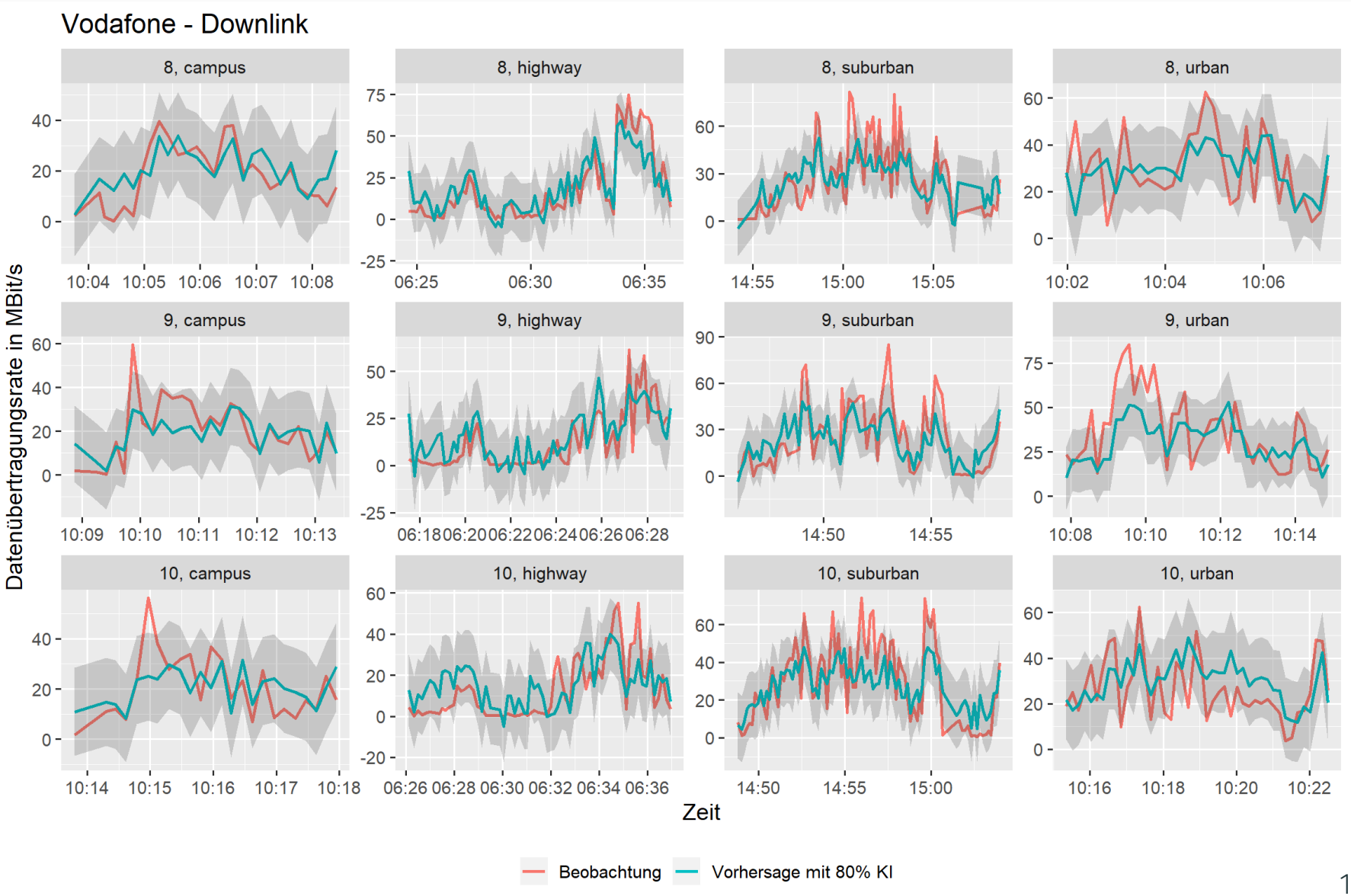
Regression mit ARMA-Fehlern: Ergebnisse (Uplink)



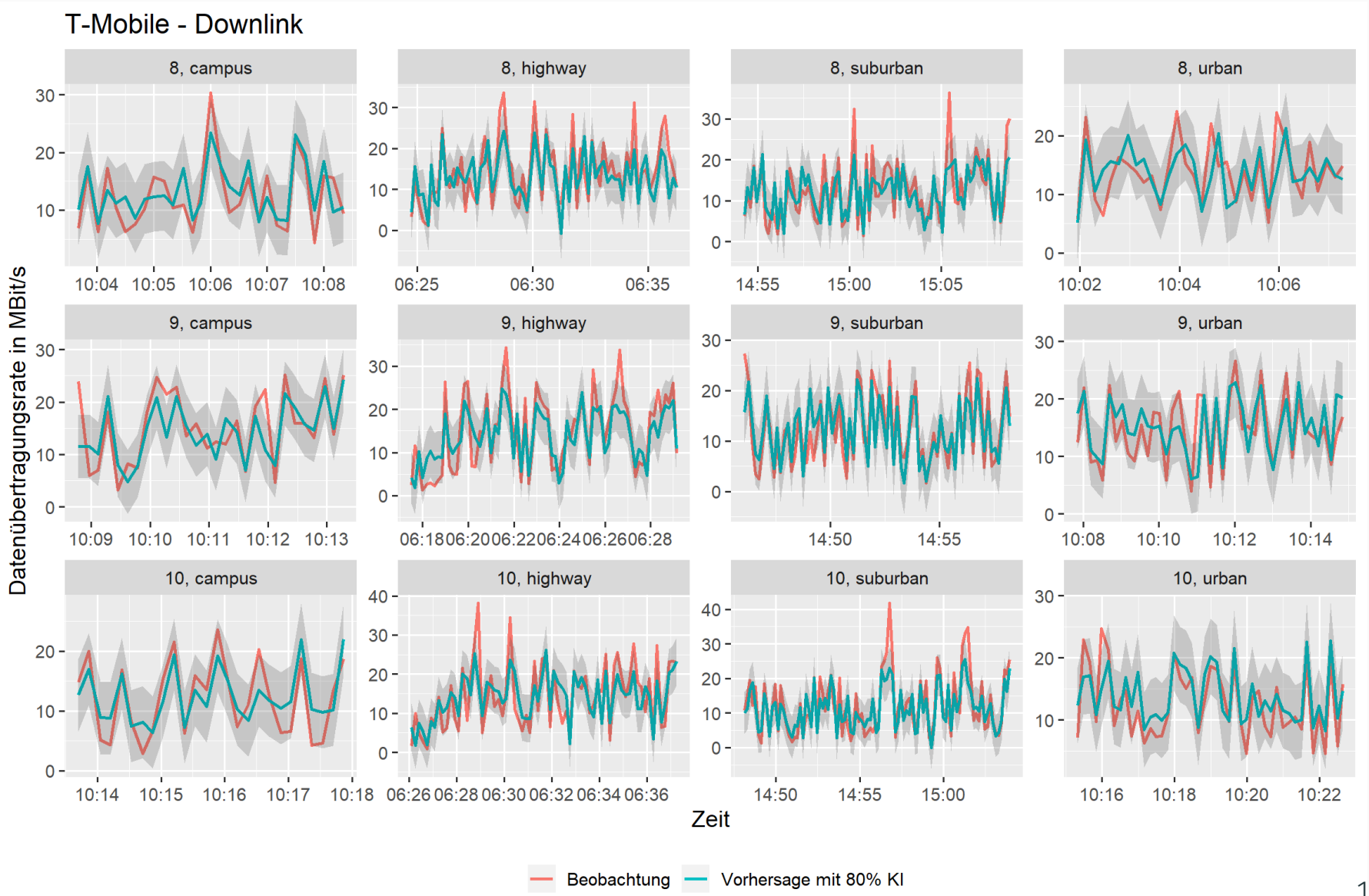
Regression mit ARMA-Fehlern: Ergebnisse (Uplink)



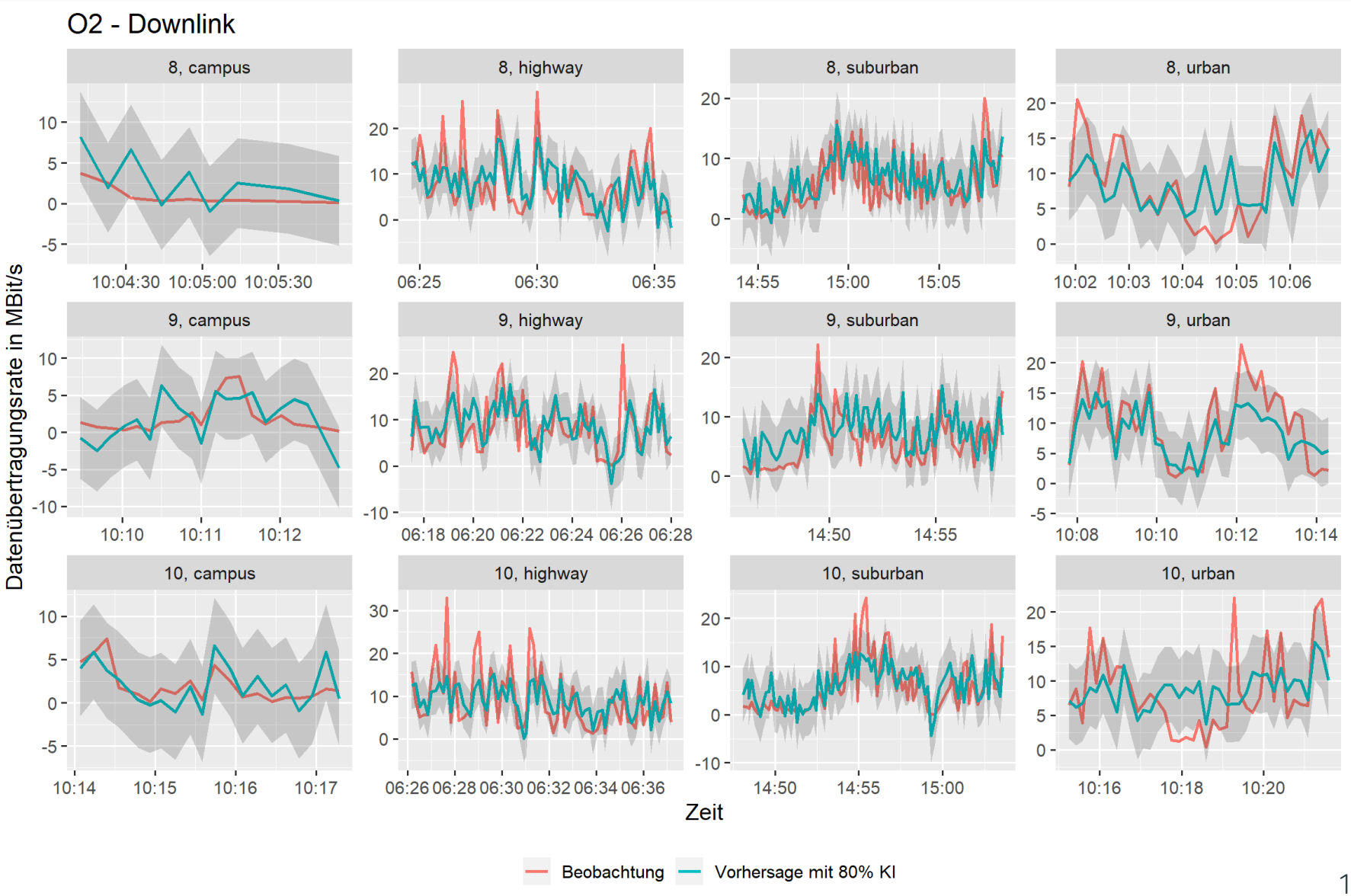
Regression mit ARMA-Fehlern: Ergebnisse (Downlink)



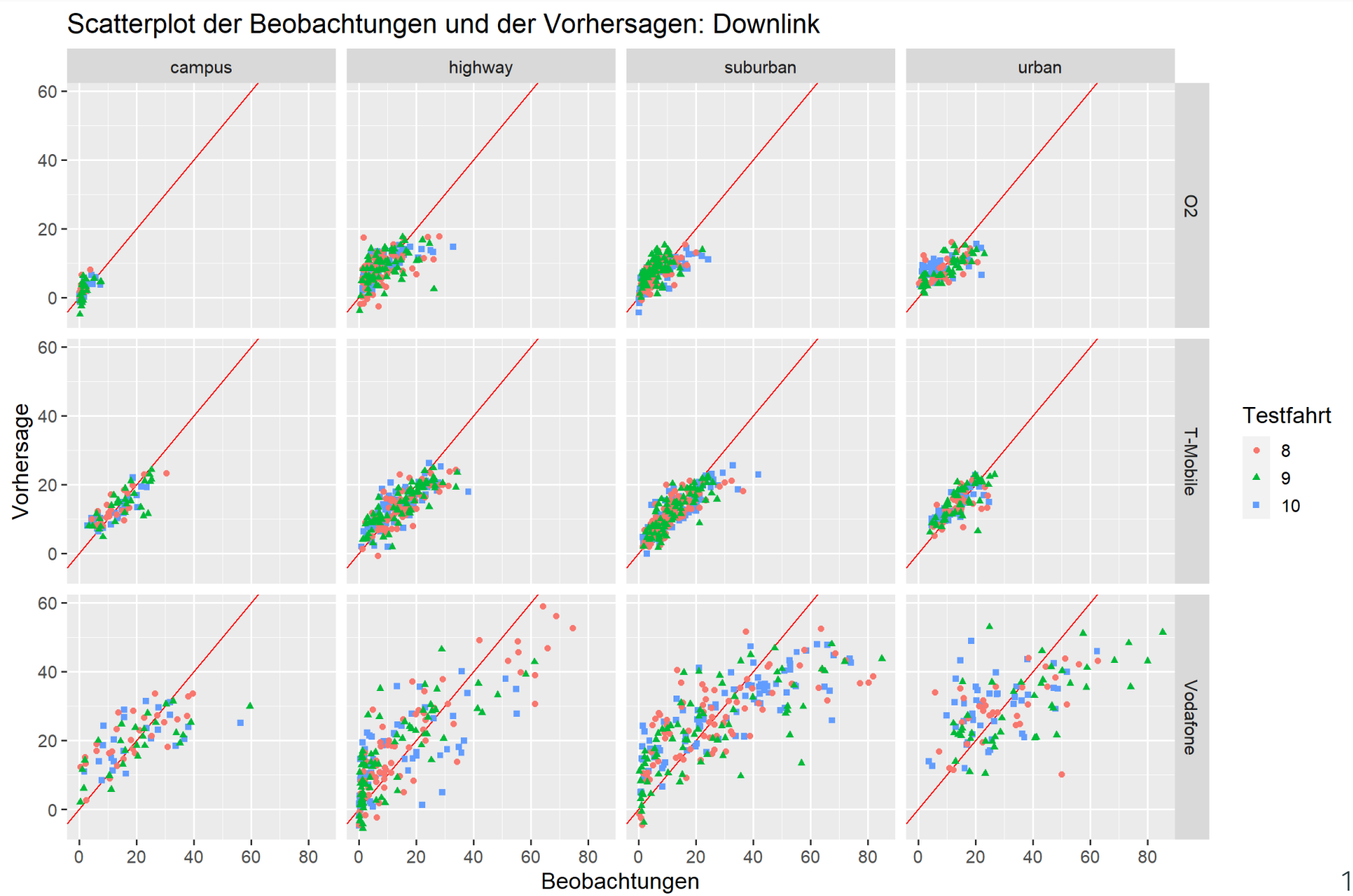
Regression mit ARMA-Fehlern: Ergebnisse (Downlink)



Regression mit ARMA-Fehlern: Ergebnisse (Downlink)



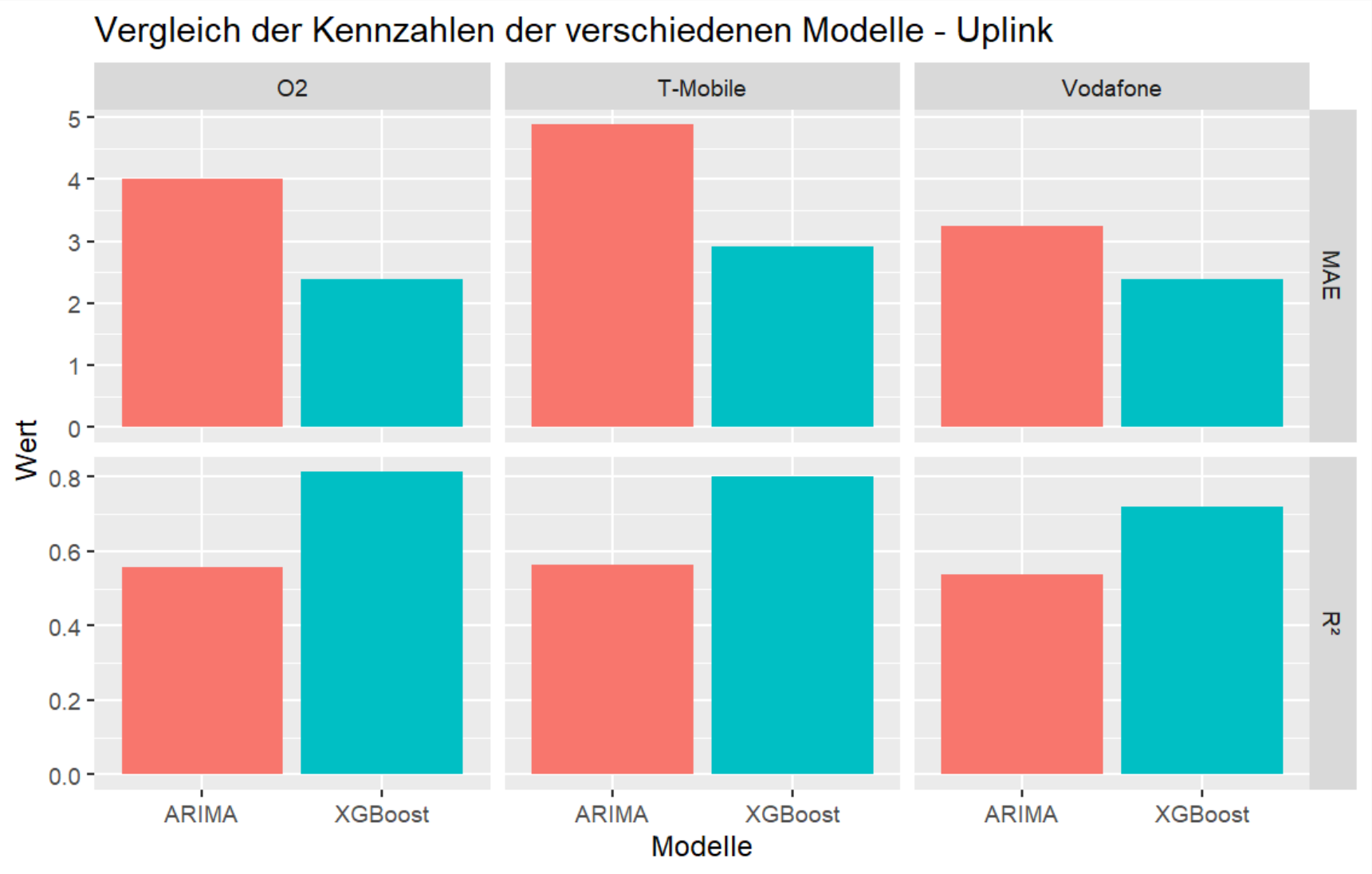
Regression mit ARMA-Fehlern: Ergebnisse (Downlink)



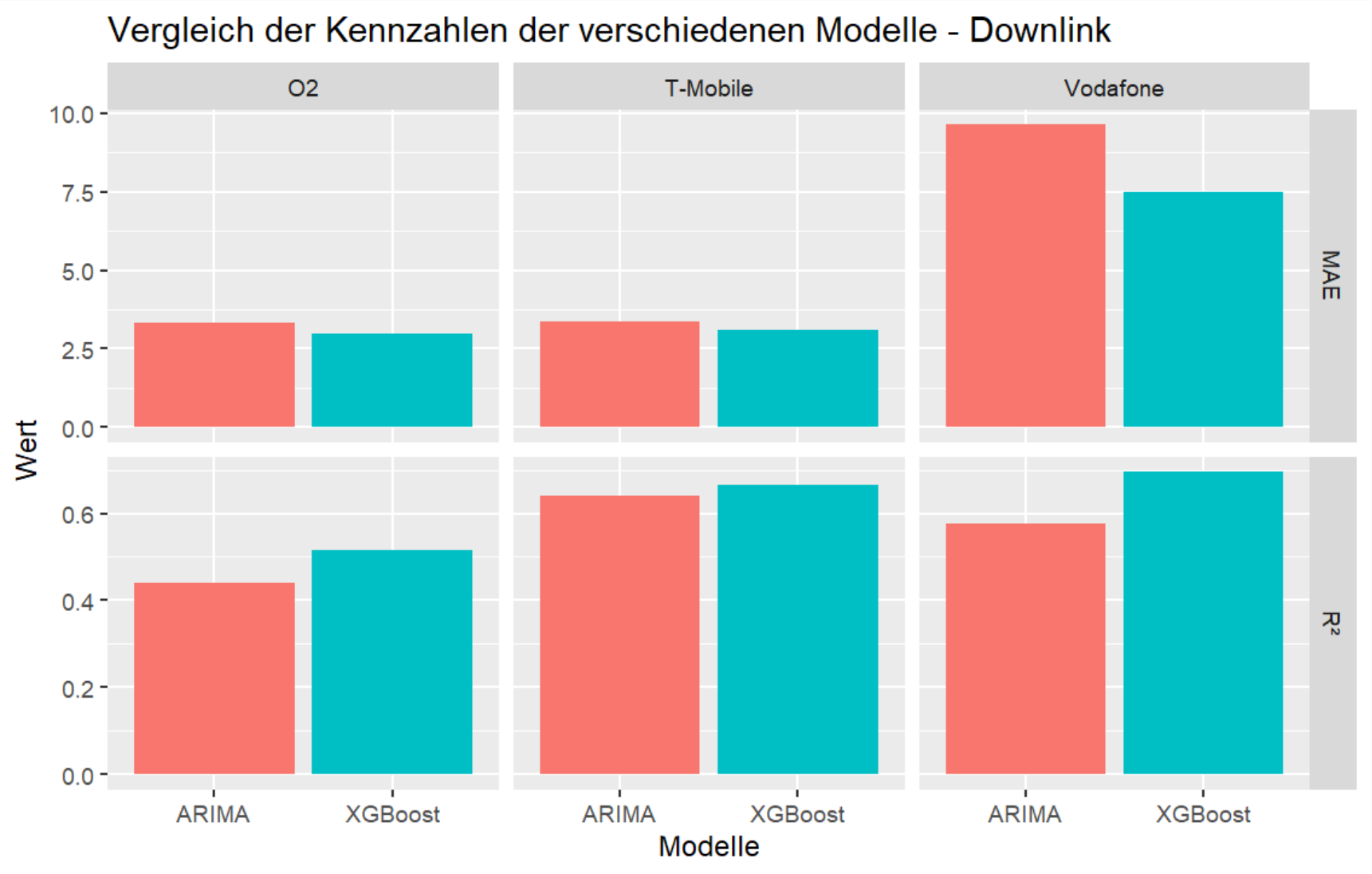
Task I - Vorhersage der Datenrate

Modellvergleich

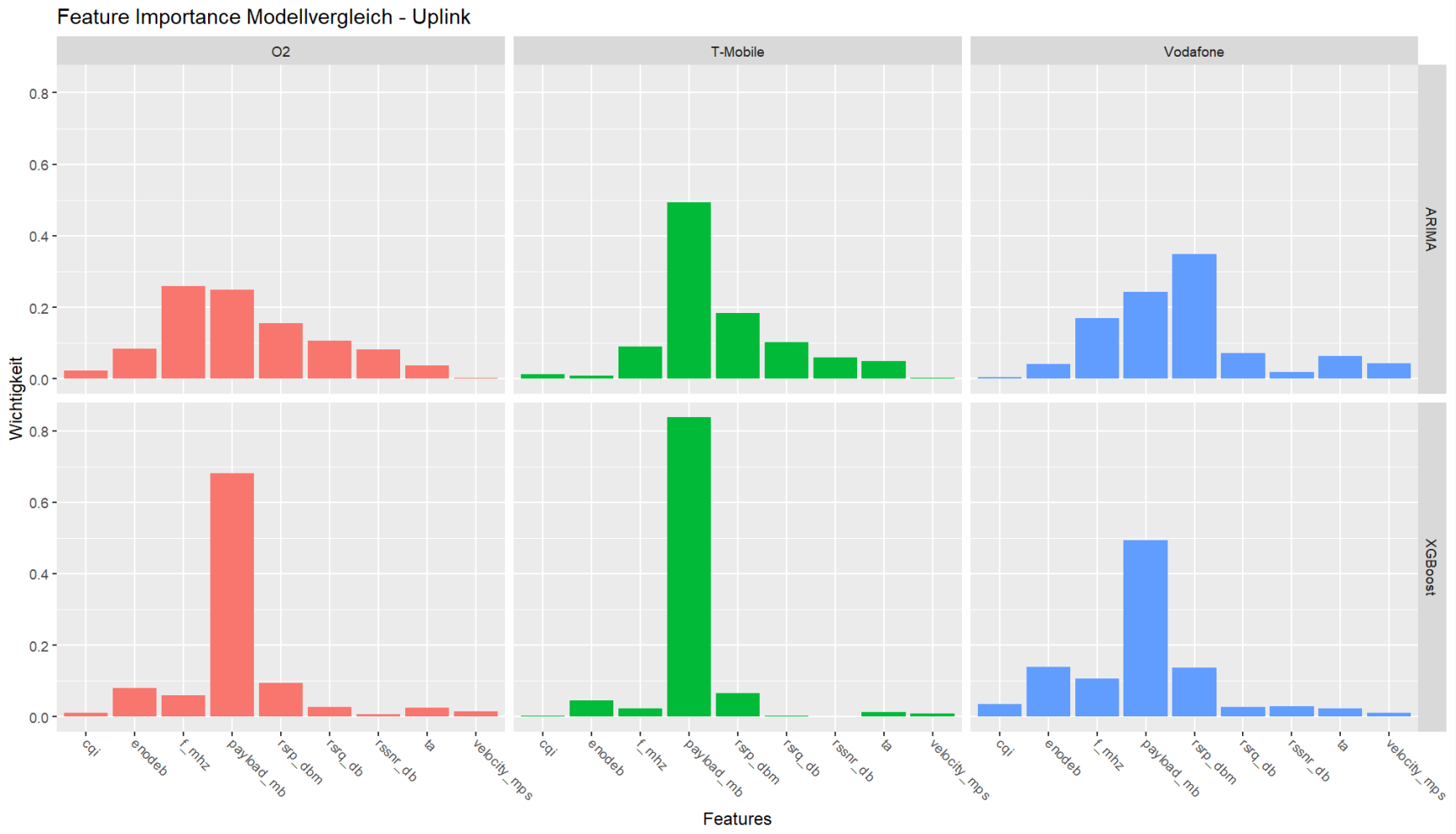
Modellvergleich Uplink - Kennzahlen



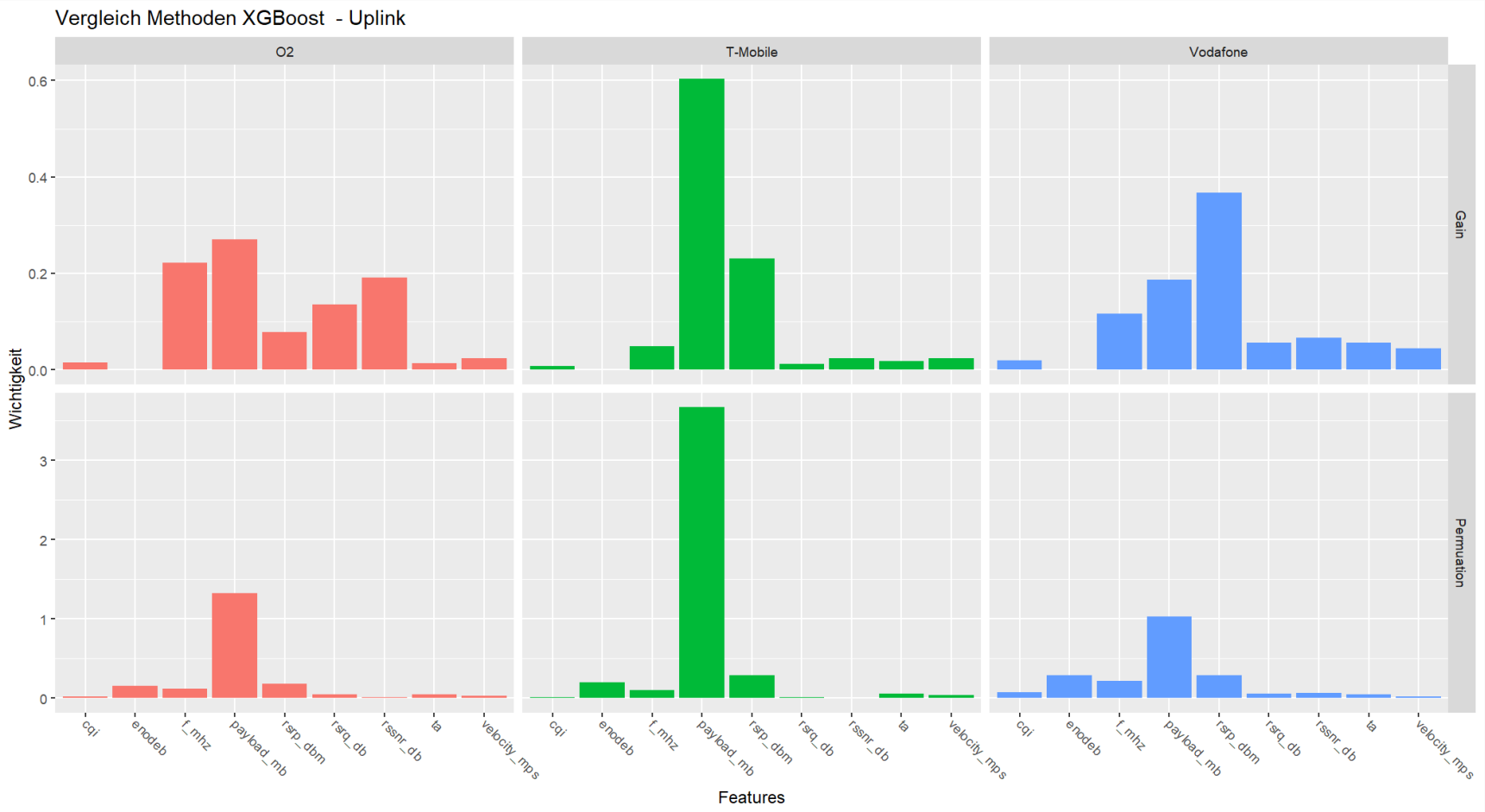
Modellvergleich Downlink - Kennzahlen



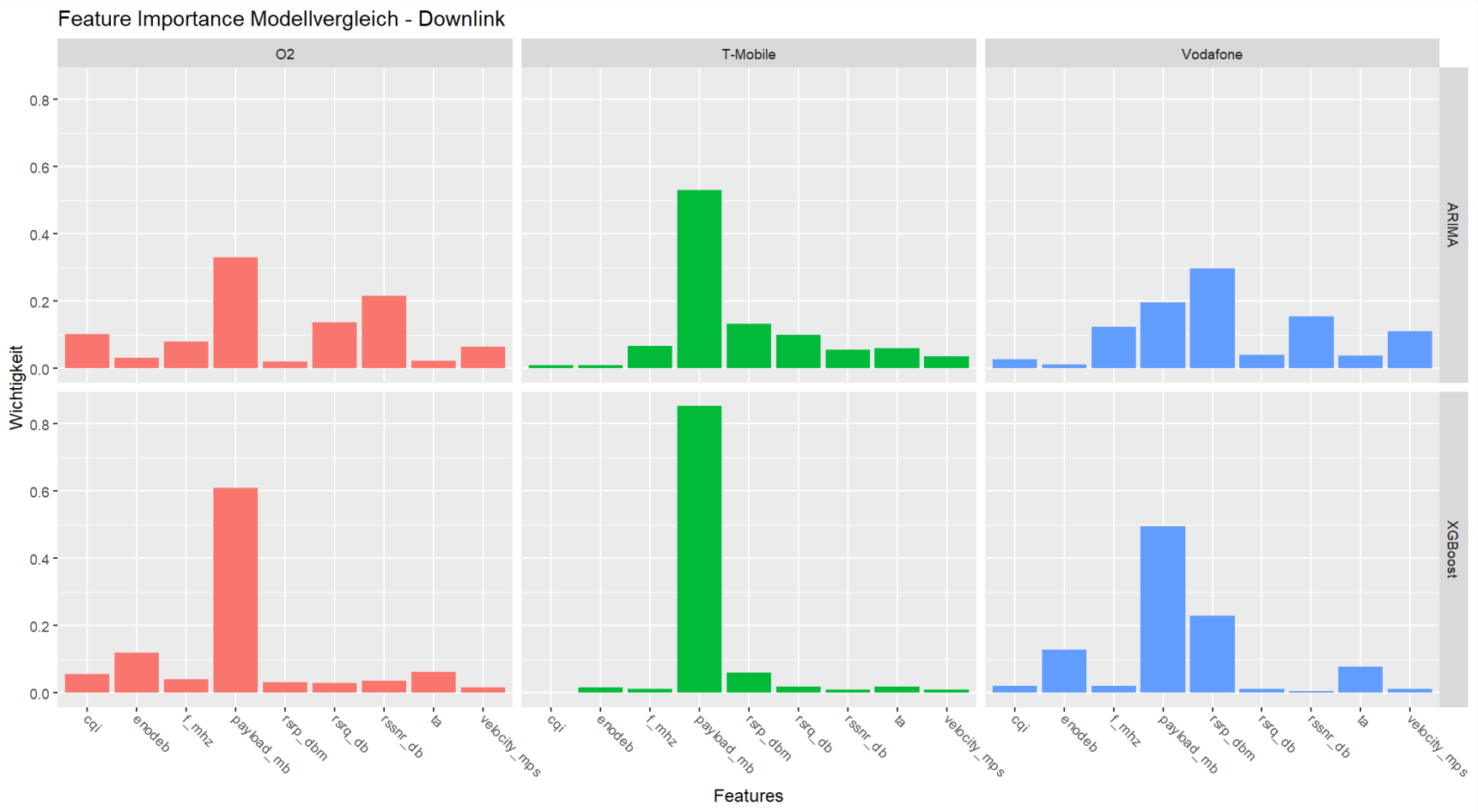
Modellvergleich Uplink - Feature Importance



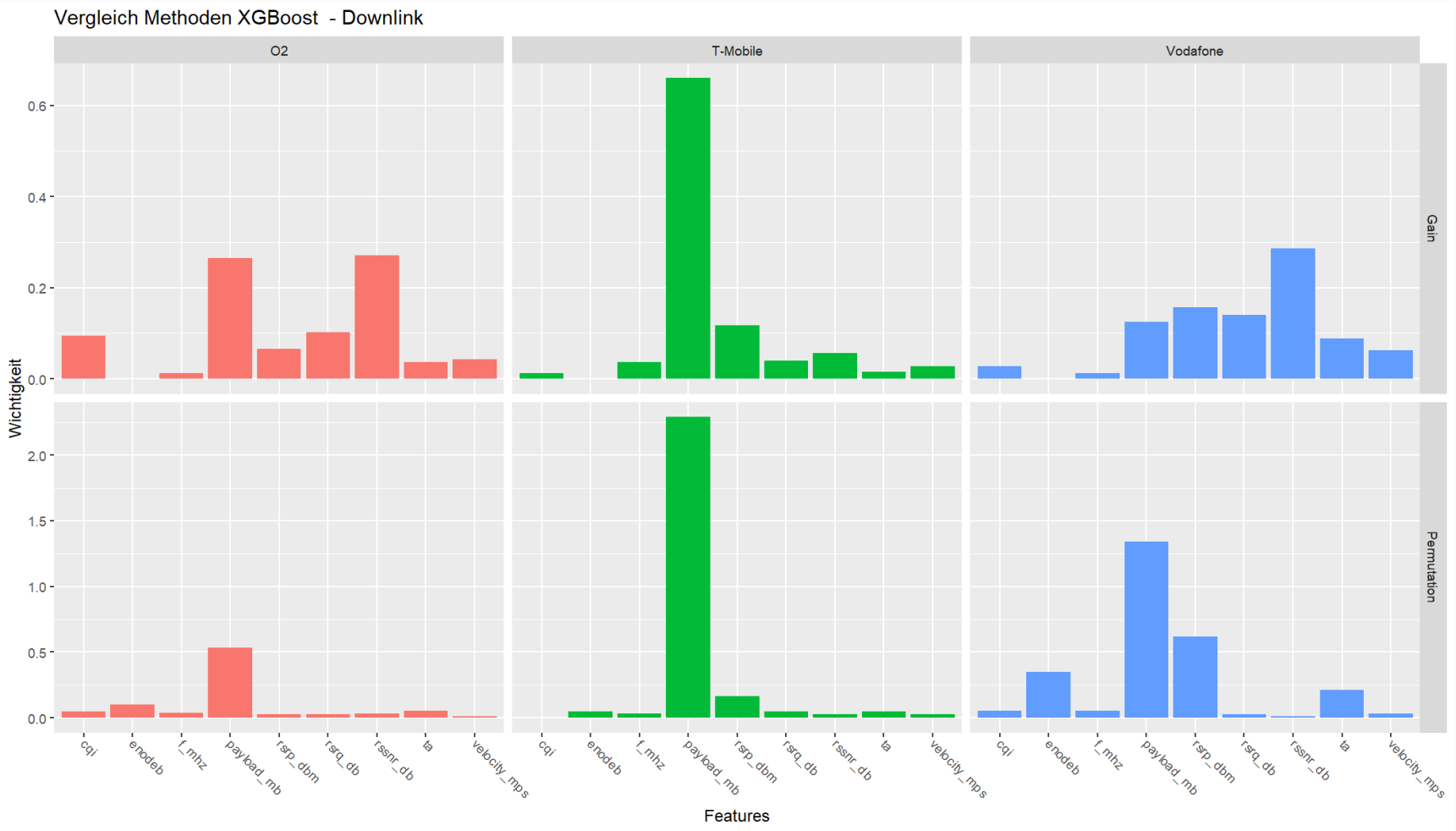
Methodenvergleich XGBoost - Uplink



Modellvergleich Downlink - Feature Importance



Methodenvergleich XGBoost - Downlink



Datentransformation

Idee: Prädiktionsmodell XGBoost für Link Lifetime mit Einfluss des RSRP/RSRQ der verbundenen sowie der Nachbarzellen

$\rightarrow$ Datentransformation

- RSRP/RSRQ Nachbarzellen :
  $\rightarrow$ mehrere Messungen - Filtern des besten Wertes zum aktuellen Zeitpunkt
  $\rightarrow$ keine Messungen - Übernehmen des letzten Wertes
- eNodeB Wechsel $\rightarrow$ Response Variable Link Lifetime

Features

- **link_lifetime** : Link-Lifetime
- **rsrp_dbm/rsrq_db** : Signalstärke/Signalqualität (RSRP/RSRQ) der verbundenen Zellen
- **rsrp_neighbor/rsrq_neighbor** : Signalstärke/Signalqualität (RSRP/RSRQ) der Nachbarzellen
- **rssnr_db** : Signal-Rausch-Verhältnis (RSSNR)
- **eNodeB** : Funkmasten im LTE-Netzwerk
- **velocity_mps** : Geschwindigkeit des mobilen Endgeräts
- **ta** : Timing Advance (TA) - Wert zur Synchronisation zwischen Up- und Downlink
- **cqi** : Channel Quality Indicator (CQI)

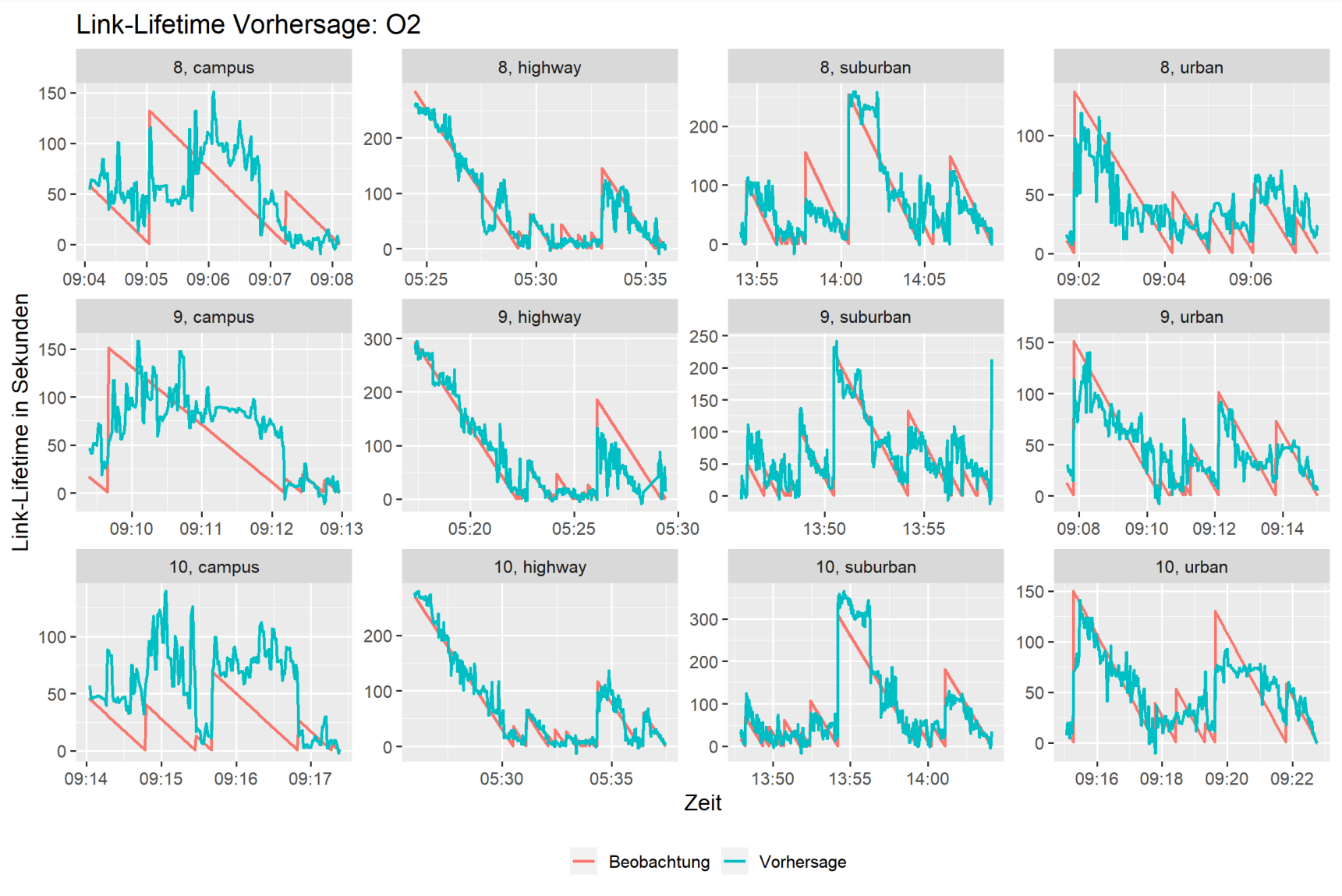
Vorgehen

Wichtige Schritte :

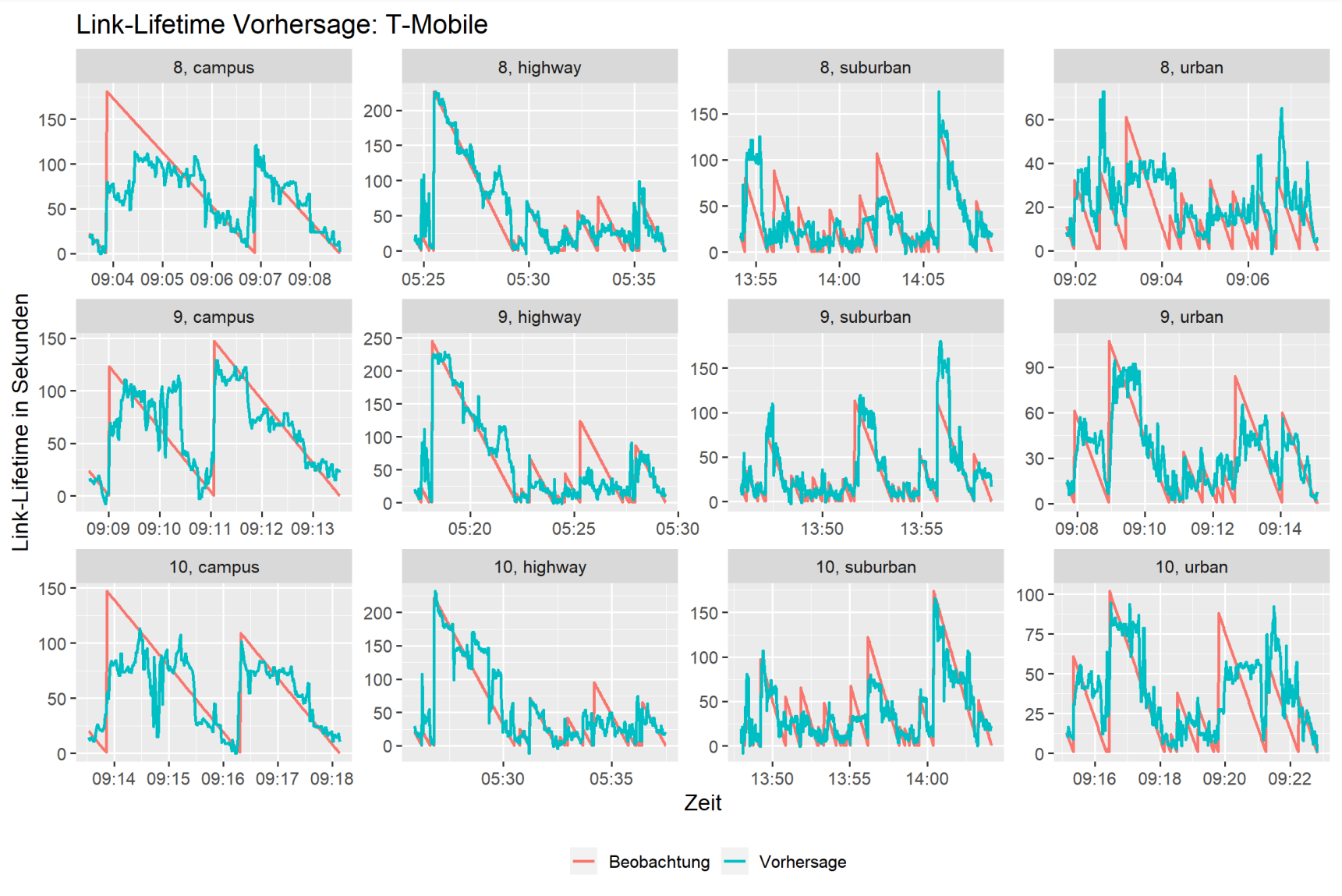
- Aufsplitten der Daten - Training/ Test
- Zufälliger Grid-Search
- Tunen der Parameter - Zeitreihenkreuzvalidierung
- Validieren des Modells auf dem Testdatensatz

$\rightarrow$ **Analog zu Task I**

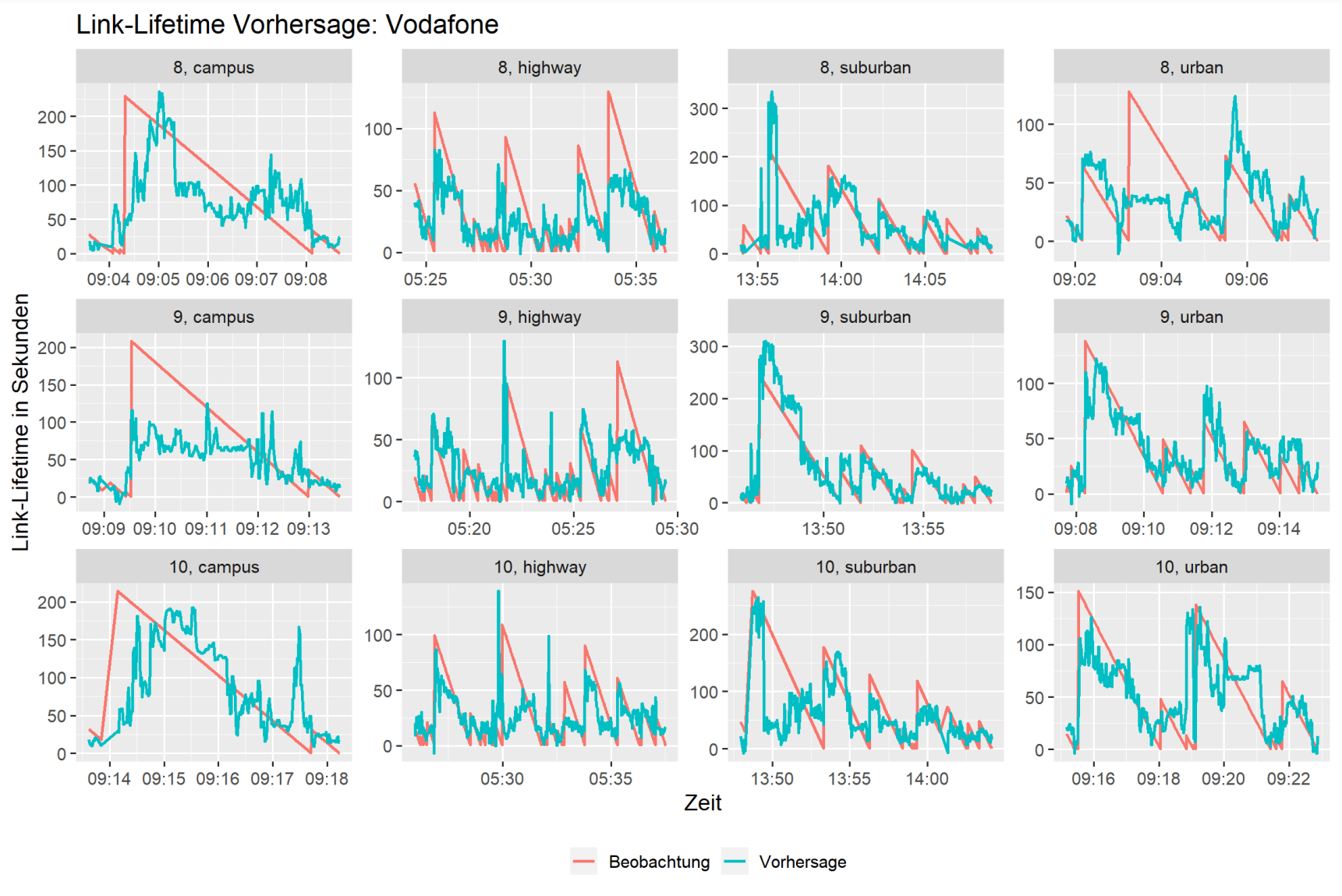
Ergebnisse - Zeitreihenplot O2



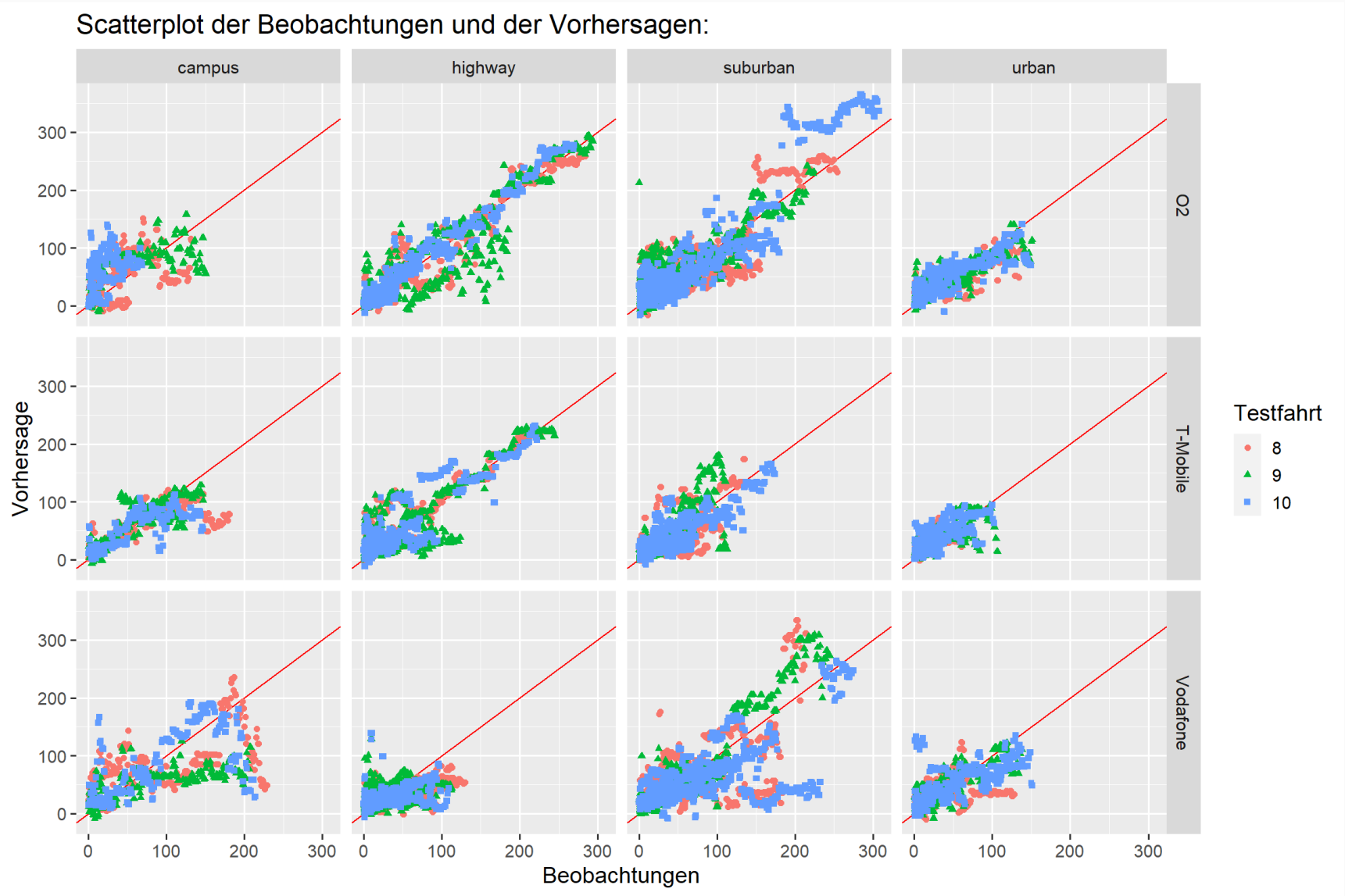
Ergebnisse - Zeitreihenplot T-Mobile



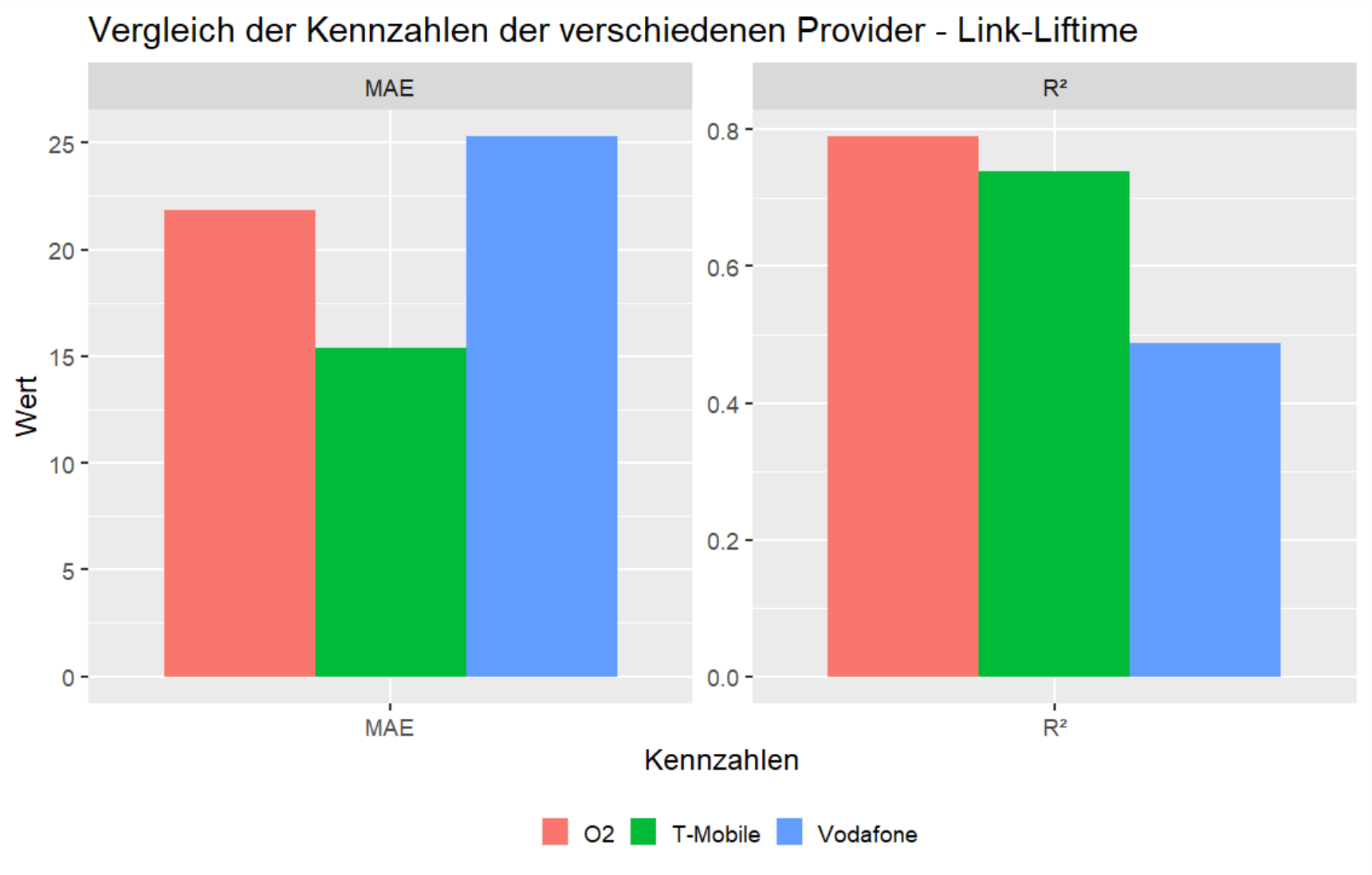
Ergebnisse - Zeitreihenplot Vodafone



Ergebnisse - Scatterplot



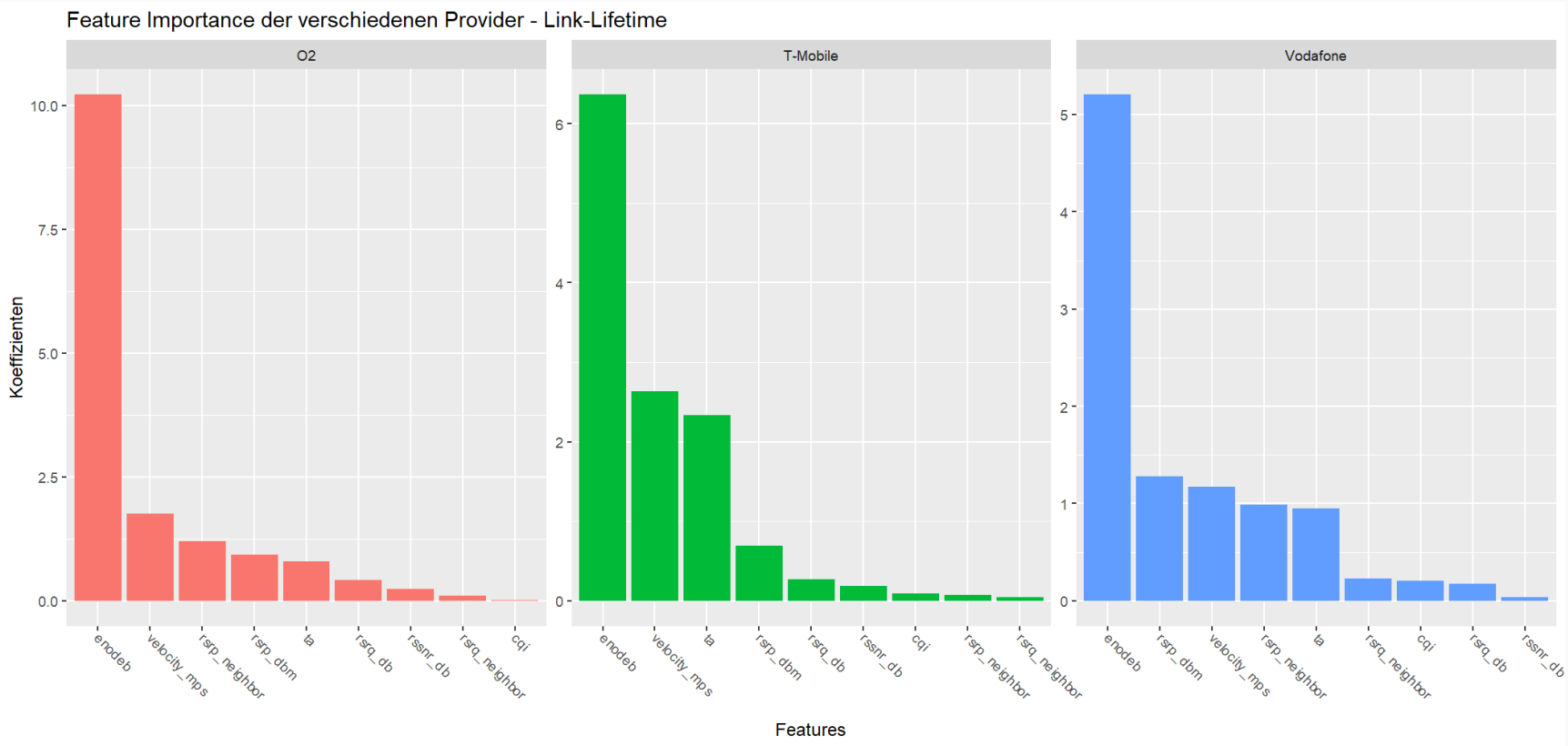
Ergebnisse - Kennzahlen



Task II - Handover Vorhersage und Link Lifetime

Feature Importance

Feature Importance



Ausblick

Ausblick

Verbesserung des Tuning-Verfahrens

- Latin Hypercube Sampling statt fixes Gitter
  - Mehr Diversität innerhalb der Parameter trotz gleichmäßiger Abdeckung des Suchraumes
- Black-Box Optimization wie z.B. Evolutionäre Algorithmen anstelle von Gittersuche

Sensitivitätsanalyse der Hyperparameter

- Welche Parameter machen wirklich einen Unterschied?

Literatur i


T. Chen and C. Guestrin.
Xgboost: A scalable tree boosting system.
CoRR, abs/1603.02754, 2016.


T. Hastie, R. Tibshirani, and J. Friedman.
The elements of statistical learning: data mining, inference and prediction.
Springer, 2 edition, 2009.


R. Hyndman and G. Athanasopoulos.
Forecasting: principles and practice, 2018.


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

Gradient Boosted Trees

- Kann man aus vielen „schwachen “ Lernern einen starken Lerner konstruieren?
  - $\Rightarrow$ Ja, Boosting ist eines der mächtigsten Konzepte des Machine Learning [2]
- Kombination von einfachen CART Bäumen zu einem starken Ensemble
  - $\Rightarrow$ Ähnlich zu Random Forest
- Der Unterschied zum Random Forest liegt im Training!

Training von Gradient Boosted Trees

- Bäume werden nacheinander zum Ensemble hinzugefügt
- Jeder neue Baum versucht, die Schwächen seiner Vorgänger äuszubügeln”
  $\Rightarrow$ *Additives Training*
- Je mehr Bäume aufgenommen werden, desto geringer wird der Training-Error (das Modell wird aber komplexer)
  $\Rightarrow$ Kontrolle des *Bias-Variance Tradeoffs*
  $\Rightarrow$ Zusätzlich gibt es Regularisierungs-Parameter

Implementierung: XGBoost

- Liefert state-of-the-art Performance in einer Vielzahl von ML-Problemen
- In 2015 haben 19/25 Gewinner von Kaggle-Competitions XGBoost eingesetzt
- Kann problemlos auf mehrere Milliarden Training Samples skaliert werden
- Lässt sich aber auch hervorragend auf ressourcenbegrenzten Systemen einsetzen [1]