Pipeline für ein künstliches neuronales Netzwerk

Aufgabe:

Der Überlebensstatus (Target “Survived“) der Passagiere auf der Titanic soll anhand der Features aus dem Kaggle-Datensatz Titanic vorhergesagt werden. Da das Target “Survived“ eine boolsche Variable ist, liegt hier eine Klassifikationsaufgabe vor.

Zielsetzung:

Zur Vertiefung des Wissen zu Deep Learning Modellen und zu Datenvorverarbeitung aus den Machine-Learning-Kaggle-Kursen, soll hier zur Vorhersage ein neuronales Netzwerk (TensorFlow) eingesetzt werden, welches mit einer eigens entwickelten Datenvorverarbeitungs-Pipeline gespeist wird.
Zur Realisierung der Datenvorverarbeitungs-Pipeline soll hier eine Bibliothek mit Datenvorverarbeitungsfunktionen entwickelt werden, welche man in einer Pipeline zusammenfassen kann. Da die Bibliothek scikit-learn keine Pipeline-Funktionen für neuronale Netzwerke aus dem TensorFlow-Framework bietet, wird die Datenvorverarbeitungs-Bibliothek in Pandas implementiert.

Zur Optimierung der Vorhersagegenauigkeit und der Stabillität des Trainingsverlaufes ist die Architektur des neuronalen Netzwerkes sowie die Trainingseinstellungen zu optimieren.

Vorgehensweise:

Folgende Arbeitsschritte werden in dieser Reihenfolge umgesetzt:

1. Entwicklung der Datenvorverarbeitungs-Bibliothek und eine daraus automatisierte Datenvorverarbeitungspipeline

2. Erstellen einer ersten Architektur des neuronalen Netzwerkes sowie Festlegung der Trainingseinstellung

3. Training des neuronalen Netzwerk mit vorverarbeiteten Daten aus der automatisierten Pipeline sowie Darstellung des Trainingsverlaufes

4. Optimierung der Architektur des neuronalen Netzwerkes sowie der Trainingseinstellung bezüglich eines stabilen Trainingsverlaufes und einer höheren Vorhersagegenauigkeit

5. Überarbeitung der Datenvorverarbeitungs-Bibliothek zur Realisierung von Feature Engineering

6. Training des neuronalen Netzwerkes mit den neu vorverarbeiteten Daten sowie Darstellung des Trainingsverlaufes

Ergebnisse:

Erster Entwurf der Datenvorverarbeitung:

Generell soll eine Datenvorverarbeitung folgende Vorverarbeitungsschritte enthalten:
- Inferenz und Konvertierung der Feature-Datentypen
- Imputation und Encoding der Features
- Skalierung der Features

Mit der ersten Version der Datenvorverarbeitungs-Bibliothek wird eine automatisierte Pipeline realisiert, welche ohne Vorwissen des Nutzers über den Datensatz eine Vorverarbeitung durchführt. Die Verarbeitungsregeln lauten wie folgt:
- Entfernung von Features mit zu hoher Anzahl von fehlenden Werten
- Entfernung von kategorischen Features mit zu hoher Kardinalzahlen
- Imputation von fehlenden Werten anhand des Feature-Datentyps mit Ergänzung von Indikatorspalten bei numerischen Features
- One-Hot-Encoding von allen kategorischen Features
- Skalierung von Features vom Datentyp Float

Nach diesen Regeln entfernt die automatisierte Pipeline die Features:
- "Ticket" und "Name" aufgrund zu hoher Kardinalzahlen (>15)
- "Cabin" aufgrund zu hohen Anteils fehlender Werte (>50%)

Entwurf der Modellarchitektur des neuronalen Netzwerkes:

Notwendig für Klassifikationsaufgabe:
- nichtlineare Aktivierungsfunktion zwischen den Schichten
- Sigmoid-Funktion als Aktivierungsfunktion bei der Ausgangsschicht

Weitere notwendige Festlegungen
- Input der Eingangsschicht entspricht der Feature-Anzahl der vorverarbeiteten Daten

Erster Entwurf:
- 4-schichtige Architektur (Ein- und Ausgangsschicht + 2 Zwischenschichten = 4 Dense-Layers)
- jeweils eine Dropout-Schicht vor jeder Zwischenschicht zur Reduktion von Overfitting
- jeweils eine Batch-Normalization-Schicht nach jeder Zwischenschicht zur Stabilisierung des Trainings
- ReLu-Funktion als Aktivierungsfunktion für Eingangs- und jede Zwischenschicht
- 64 Neuronen für jede der 4 Schichten

Erste Trainingseinstellung:

Notwendig für Klassifikationsaufgabe:
- cross-entropy-Funktion als zu minimierende Verlustfunktion beim Training

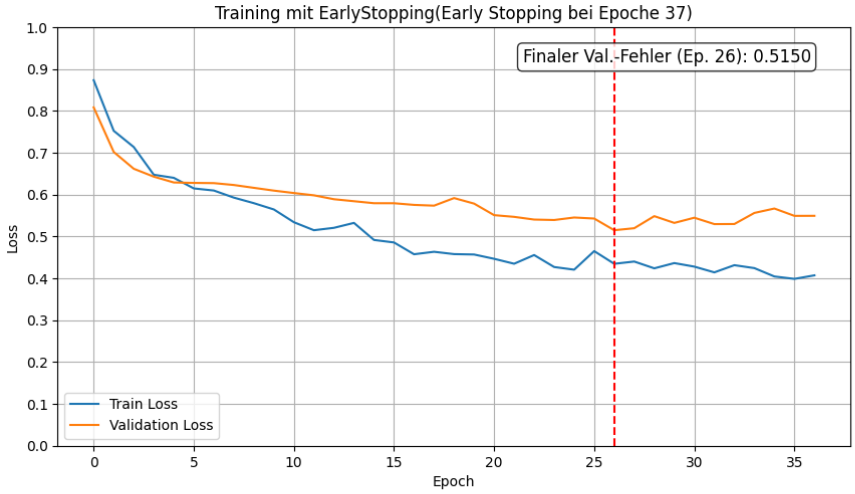
Weitere relevante Festlegungen:
- Adam als Trainingsalgorithmus für schnelles Training von komplexen Netzwerken
- Early-Stopping als Abbruchbedingung für das Training zur Vermeidung von Overfitting

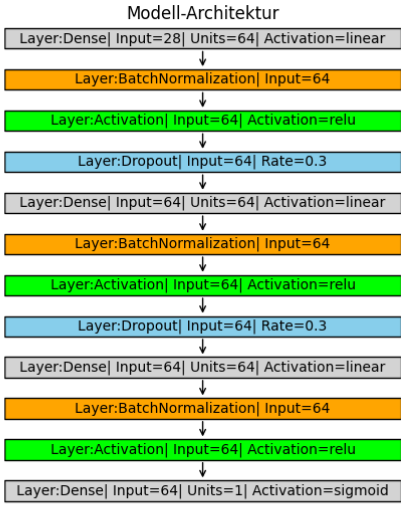
Trainingsparameter:
- Anzahl der Epochen (Epochs): 300
- Batch-Größe (Batchsize): 80
- Intervall für Early-Stopping (Patience): 10
- Differenz für Early-Stopping (min_delta): 0,01 (1%)

Zur Darstellung des Trainingsverlaufes wird hier der Verlauf des Trainings- und Validierungsfehlers aufgezeichnet. Als Maß für die Vorhersagegenauigkeit wird hier der Validierungsfehler ausgewählt. (Binary-Accuracy wird hier nicht betrachtet)

Training des neuronalen Netzwerkes:

Mit der vorher genannten Datenvorverarbeitung , Modellarchitektur und Trainingseinstellungen ergibt sich dieser Trainingsverlauf:





Parameter	Epochs (max)	Batchsize	min_delta	Patience	Optimizer	Loss-Function	Metric
Werte	300	80	0.01	10	Adam	binary_crossentropy	binary_accuracy

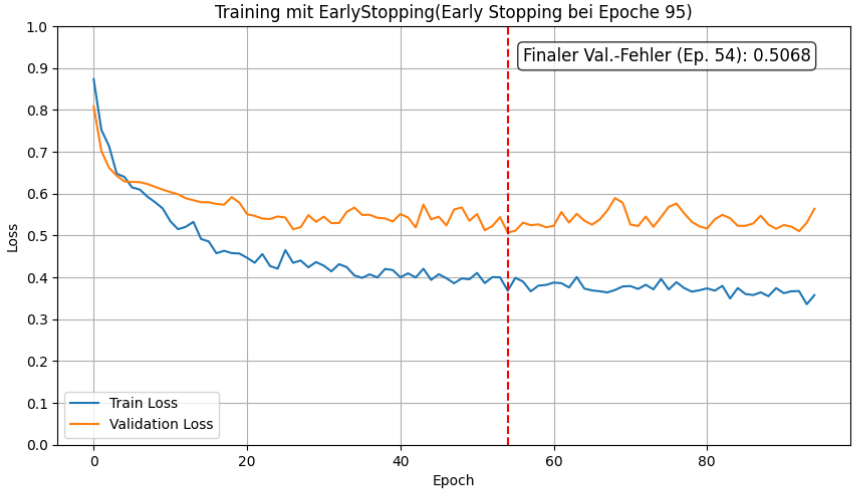
(Hinweis: 61,47% Validierungsfehler beim Ausschalten aller Batch-Normalization-Schichten!)

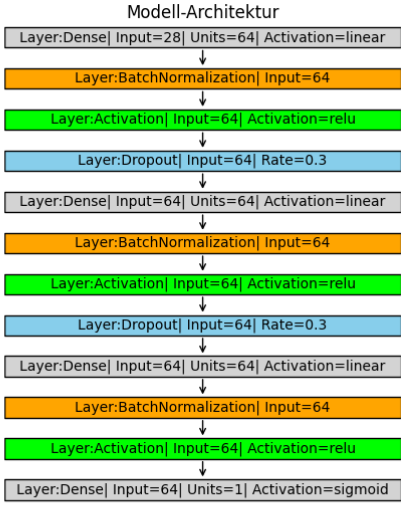
Optimierung der Modellarchitektur und Trainingseinstellung:

Bezüglich eines stabileren Trainingsverlaufes sowie eines niedrigeren Validierungsfehlers werden folgende Maßnahmen ausgehend vom ersten Entwurf ergriffen:
Strengere Abbruchbedingungen bei Early-Stopping:

- Intervall für Early-Stopping (Patience): 10 ⇒ 40
- Differenz für Early-Stopping (min_delta): 0,01 (1%) ⇒ 0,0001 (0,01%)

liefert instabileren Trainingsverlauf und dezent niedrigeren Validierungsfehler:





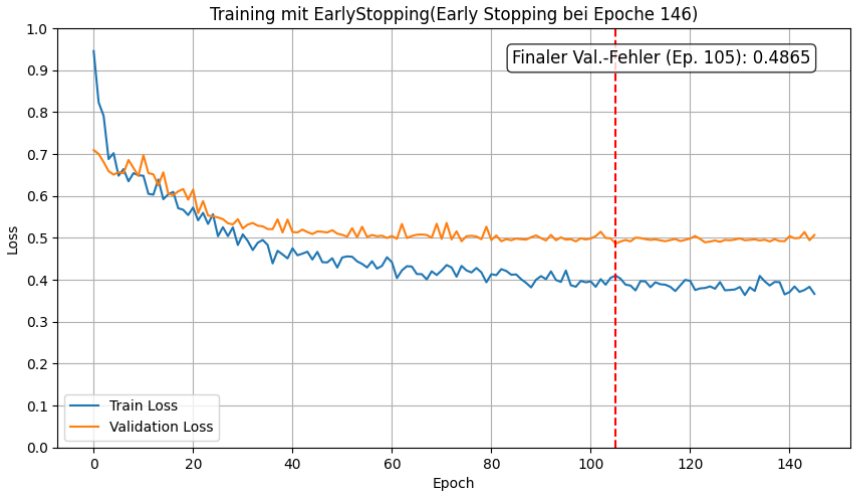
Parameter	Epochs (max)	Batchsize	min_delta	Patience	Optimizer	Loss-Function	Metric
Werte	300	80	0.0001	40	Adam	binary_crossentropy	binary_accuracy

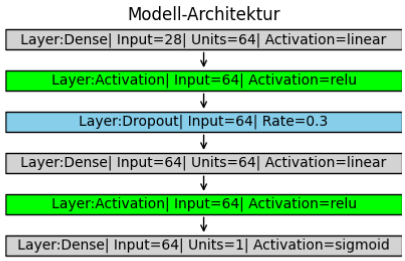
(Hinweis: 49,05% Validierungsfehler beim Ausschalten aller Batch-Normalization-Schichten)

Strengere Abbruchbedingungen bei Early-Stopping + Entfernung von Zwischenschichten:

- Entfernen von Zwischenschichten sowie die dazugehörigen Batch-Normalization- und Dropout-Schichten

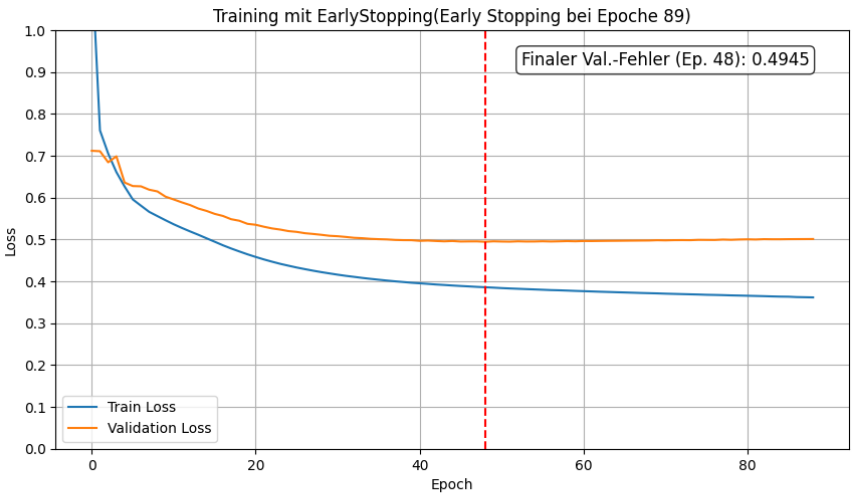
stabilisiert den Trainingsverlauf und reduziert den Validierungsfehler.
Der niedrigste Validierungsfehler wird bei einer Architektur mit einer Zwischenschicht und ohne Batch-Normalization-Schichten erreicht:

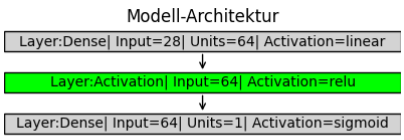




Parameter	Epochs (max)	Batchsize	min_delta	Patience	Optimizer	Loss-Function	Metric
Werte	300	80	0.0001	40	Adam	binary_crossentropy	binary_accuracy

Der stabilste Trainingsverlauf wird bei einer Architektur ohne Zwischen- und Batch-Normalization-Schichten erreicht:





Parameter	Epochs (max)	Batchsize	min_delta	Patience	Optimizer	Loss-Function	Metric
Werte	300	80	0.0001	40	Adam	binary_crossentropy	binary_accuracy

Wahl der besten Architektur:
Eine ausreichende Stabilisation des Trainingsverlaufes durch Variation der Batchgröße (Batchsize) und Anzahl der Neuronen, bei der Modellarchitektur die den niedrigsten Validierungsfehler liefert, konnte nicht erreicht werden. Folglich fällt die Wahl für die optimale Modellarchitektur auf das Modell ohne Zwischen- und Batch-Normalization-Schichten. Dadurch nimmt man hier einen geringfügig höheren Validierungsfehler (49,45% zu 48,65%) für eine erhebliche Stabilisierung des Trainingsverlaufes in Kauf.

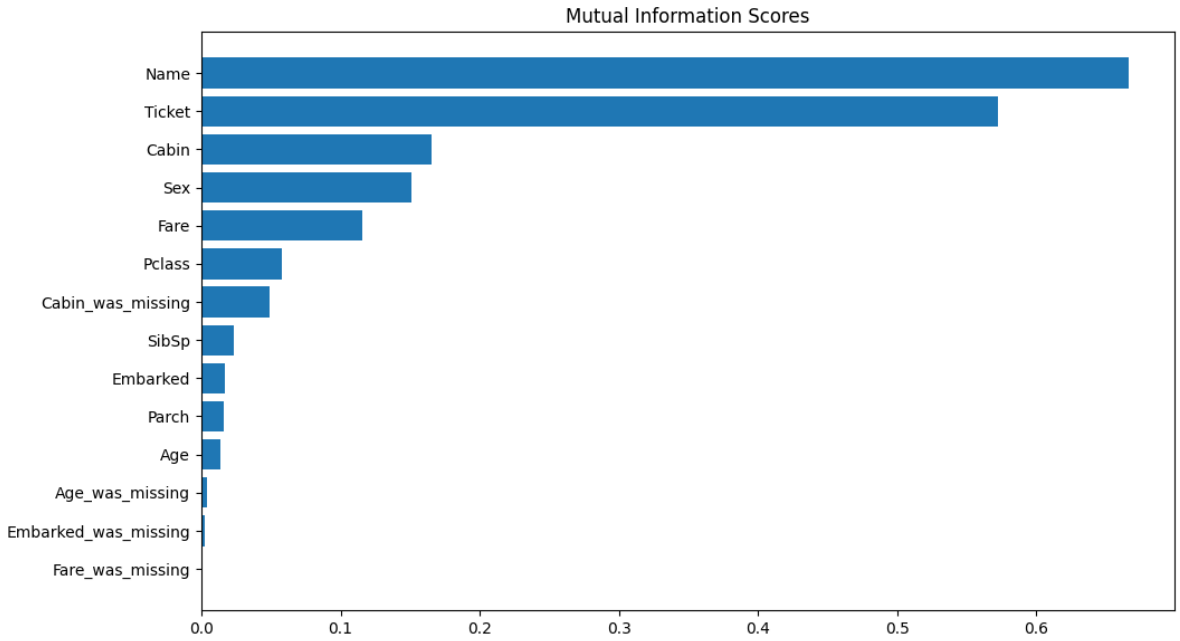
Überarbeitung der Datenvorverarbeitung und Realisierung von Feature-Engineering:

Überarbeitung der Datenvorverarbeitung:
Starke Kapselung der Datenvorverarbeitungsschritte in einzelne Funktionen

Verbesserung zum ersten Entwurf der Datenvorverarbeitungs-Bibliothek:
- erheblich höhere Übersichtlichkeit und Erweiterbarkeit
- Möglichkeit für Feature Engineering
- Einführung von Embedding-Encoding für kategorische Features
- flexiblere Wahl von Encoding-Möglichkeiten

Unterschiede zum ersten Entwurf der Datenvorverarbeitungs-Bibliothek:
- Imputation von fehlenden Werten erfolgt immer mit der Ergänzung einer Indikatorspalte

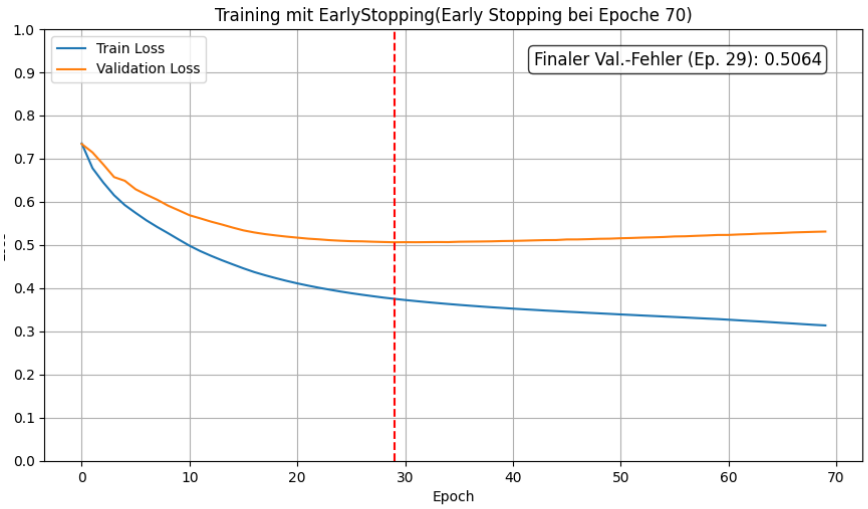
Mit der neuen Datenvorverarbeitung lässt sich nach der Inferenz und Konvertierung der Feature-Datentypen sowie der Imputation der Features die Mutual-Information berechnen:

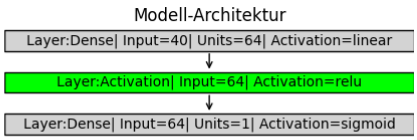


Demnach lässt sich aus den Features “Name“ und “Ticket“ am meisten von allen Features auf das Target “Survived“ schließen! Demnach haben diese beiden Features einen höheren Einfluss auf die Vorhersagegenauigkeit und sollten beim Training deshalb mit einbezogen werden.
Diese beiden kategorischen Features lassen sich hier mit Embedding-Encoding behandeln anstatt diese aus dem Datensatz zu entfernen wegen ihrer hohen Kardinalzahlen wie bei der automatisierten Pipeline. Ansonsten erfolgt das Encoding und Skalieren wie bei der automatisierten Pipeline.

Training des neuronalen Netzwerkes mit den neu vorverarbeiteten Daten:

Mit der obigen Datenvorverarbeitung und der optimalen Modellarchitektur sowie Trainingseinstellung, aus der Optimierung, erhält man:





Parameter	Epochs (max)	Batchsize	min_delta	Patience	Optimizer	Loss-Function	Metric
Werte	100	80	0.0001	40	Adam	binary_crossentropy	binary_accuracy

und damit einen geringfügig höheren Validierungsfehler (50,64% zu 49,45%) als ohne Einbeziehung der Features “Name“ und “Ticket“ wie bei der automatisierten Pipeline.

Fazit:
Prinzipiell scheint hier das Training sehr instabil zu sein, da der hier verwendete Datensatz wahrscheinlich zu klein für ein neuronales Netzwerk ist. Stabilisierend wirkte sich hier das Ausschalten von Batch-Normalization-Schichten aus (entgegen des eigentlichen Zwecks von Batch-Normalization-Schichten). Ebenfalls kann das Training stabilisiert werden, indem Zwischenschichten ausgeschaltet werden (weniger tiefe Netzwerk-Architektur).

Prinzipiell lässt sich der Validierungsfehler nur geringfügig durch Variation der Trainingseinstellung und Modellarchitektur verbessern. Ausnahme bildet hier die Modellarchitektur mit zwei Zwischenschichten bei lockerer Abbruchbedingung bei Early Stopping (51,50% mit Batch-Normalization-Schichten zu 61,47% ohne).

Die Einbeziehung von Features mit höherer Mutual-Information führt zu einem höheren statt niedrigeren Validierungsfehler.

Hinweis:
Da diese Datenvorverarbeitung Data Leakage verursacht, können die Validierungsfehler verzerrt sein (zu günstig/niedrig). Daher ist bei weiteren Überarbeitungen der Datenvorverarbeitung Data Leakage zu entfernen.