Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

2024 FS CAS PML - Supervised Learning
5 Deployment und Abschluss

Werner Dähler 2024

5 Deployment und Abschluss - AGENDA

51. Das Finale Model
52. Feature Engineering in der Produktion
53. Modellübergabe in die Produktion
54. Der Modellierungsprozess
55. PyCaret

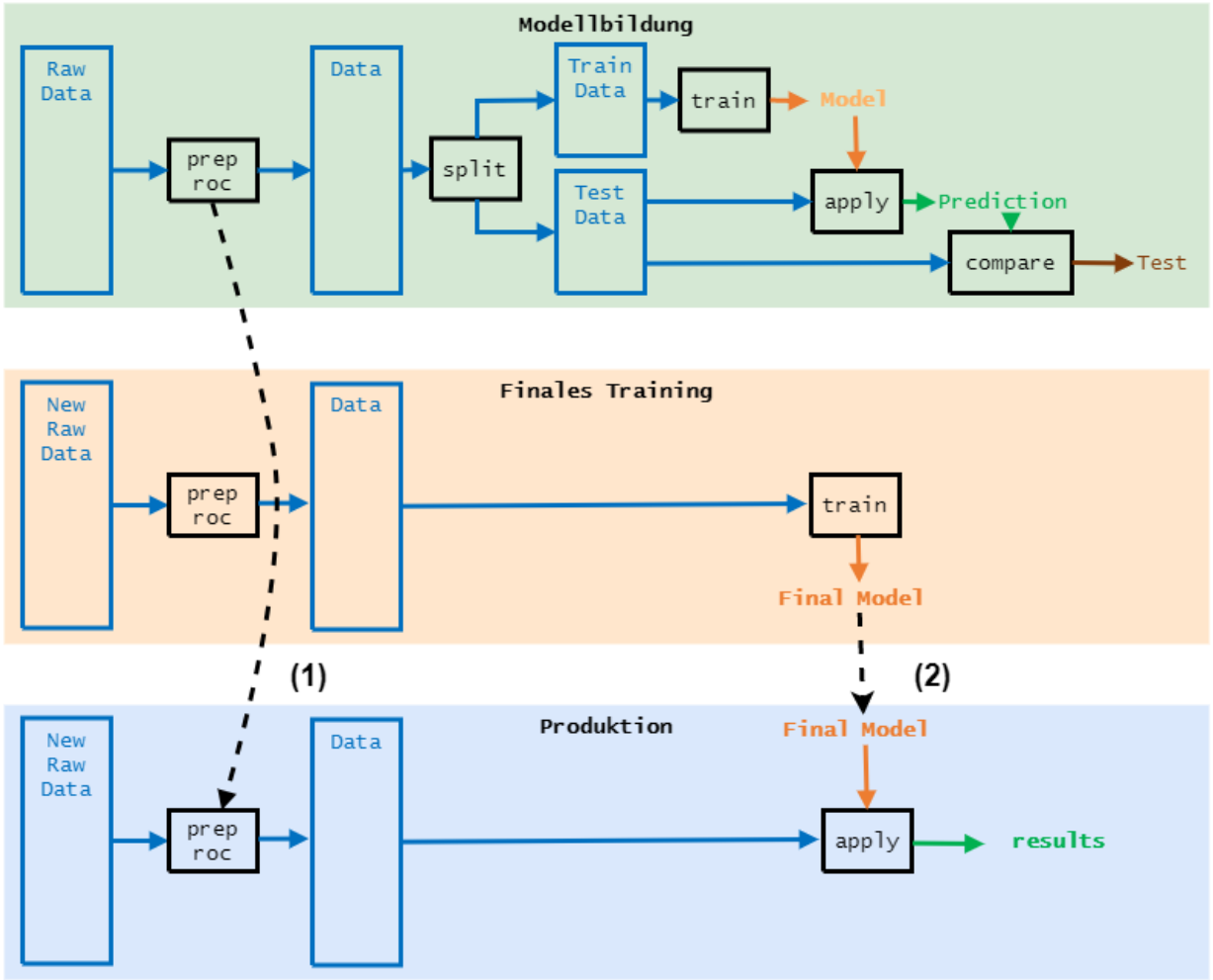
die hinterlegten Links wurden am 27.05.2024 abgegriffen

5.1 Deployment und Abschluss - Das Finale Modell

- finales Ziel von Predictive Modelling ist es, ein ausgereiftes und vom Fach akzeptiertes (erwachsenes) Modell in Produktion zu bringen

- das finale Modell wird dabei auf allen zur Verfügung stehenden Daten trainiert (also ohne Train - Test - Split)

- für die Produktivsetzung müssen dabei die folgenden Objekte vorbereitet werden
  - Feature Engineering (1)
  - das final trainierte Modell (2)



5.2 Deployment und Abschluss - Feature Engineering in der Produktion

- grundsätzlich müssen Transformationen des Feature Engineering (vgl. Kap. 1.6 Feature Engineering - Implementation) für neue Daten auf dieselbe Weise durchgeführt werden wie für die Modellentwicklung

- folgendes ist dabei zu beachten
  1. theoretisch können auf allen Features Missing Values auftreten
  2. Struktur der Feature Matrix muss mit jener übereinstimmen, mit welcher das Modell trainiert worden war

5.2 Deployment und Abschluss - Feature Engineering in der Produktion

5.2.1 Missing Values

- der Code ist dahingehend zu ergänzen, dass für alle Features eine angemessene Behandlung von Missing Values vorzusehen ist
- mögliche Strategien:
  - pauschale Behandlung aller numerischen Features, z.B. einsetzen des Medians, und für kategoriale Features einsetzen des Modalwertes
  - individuelle Strategien für **alle** Features anhand der Erkenntnisse von Feature Exploration

5.2 Deployment und Abschluss - Feature Engineering in der Produktion

5.2.2 Neue Kategorien

- in den Modellen von scikit-learn sind die Feature Namen nicht hinterlegt, sie werden über die Spaltenindices der Feature-Matrix angesprochen
- die neuen Daten müssen dieselbe Struktur aufweisen in Bezug auf
  - Anzahl und Anordnung der Features (rows)
  - Datentypen
- das Erscheinen von neuen Werten in kategorialen Features führt dazu, dass z.B. bei One Hot Encoding neue Dummy Variablen erzeugt werden
- dies führt zu veränderter Struktur der Feature Matrix, welche nicht mehr mit dem anzuwendenden Modell übereinstimmt

- mögliche Strategien:
  - sicherstellen der geforderten Struktur bei Übernahme aus Quellsystem
  - vorgeschalteter Konsistenztest bei Feature Engineering in der Produktion (inkl. protokollieren)

5.2 Deployment und Abschluss - Feature Engineering in der Produktion

5.2.3 Protokollieren

- für künftige Arbeiten an Modellen in weiteren Modellierungszyklen ist es angebracht, das Auftauchen von Ereignissen in neuen Daten zu beobachten und festzuhalten
- dies kann z.B. dadurch erreicht werden, indem der Code des Feature Engineering in der Produktion zusätzlich ein Log schreibt, welches danach entsprechend analysiert werden kann
  (vgl. rudimentäre Log Erstellung im [ipynb])

5.3 Deployment und Abschluss - Modellübergabe in die Produktion

- in den bisherigen Tätigkeiten waren die trainierten Modelle jeweils flüchtig (transistent), d.h. sie existierten nur innerhalb der jeweiligen Python Session
  (z.B. bei GridSearchCV können tausende Modelle temporär erstellt und wieder verworfen werden)
- um sie als eigenständige Objekte in eine andere Umgebung bringen zu können, müssen sie aber dauernd (persistent) hinterlegt werden können
  (es geht also ums Eingemachte)

- zwei Wege dazu sollen kurz aufgezeigt werden
  1. Modelle speichern mit pickle für internen Gebrauch (Python)
  2. Modelle speichern mit PMML für externen Gebrauch

5.3 Deployment und Abschluss - Modellübergabe in die Produktion

5.3.1 Modelle speichern intern mit pickle

- die Python Library pickle bietet die Möglichkeit, Python Objekte in einem binären Format als Memory Dump
  - permanent auf der Disk zu speichern ...
  - ... und von dort auch wieder zu laden
- im korrespondierenden [ipynb] werden auf Daten des Melbourne Housing Dataset drei unterschiedliche Modelle trainiert
  - model_sc (Klasse: StandardScaler)
  - model_lr (Klasse: LinearRegression)
  - model_dt (Klasse: DecisionTreeRegressor)

5.3 Deployment und Abschluss - Modellübergabe in die Produktion

5.3.1 Modelle speichern intern mit pickle

- mit pickle.dump() können beliebige Python Objekte als File abgespeichert werden:

```
import pickle
with open('model_sc.pkl', 'wb') as pickle_file:
    pickle.dump(model_sc, pickle_file)
:
```

- ein Blick in das Filesystem zeigt, dass es sich dabei offensichtlich um (proprietäre) Dumps der entsprechenden Objekte handelt

- mit pickle.load() können sie von dort auch wieder geladen werden

```
with open('model_sc.pkl', 'rb') as pickle_file:
    model_sc_2 = pickle.load(pickle_file)
:
```

5.3 Deployment und Abschluss - Modellübergabe in die Produktion

5.3.2 Modelle speichern extern mit PMML (Predictive Model Markup Language)

- ein auf XML basierender Standard zum Austausch von Ergebnissen des Data Mining zwischen Applikationen unterschiedlicher Hersteller (z.B. IBM Intelligent Miner, SAS Enterprise Miner, SPSS Clementine, KNIME)
- diese bieten (meist) eine Exportfunktion im PMML-Format an, um das so codierte Ergebnis in die Systeme anderer Hersteller importieren zu können
- neben der Möglichkeit, Modelle zwischen Applikationen unterschiedlicher Hersteller austauschen zu können, bieten nach PMML exportierte Modelle die weitere Möglichkeit, die innere Struktur derartiger Modelle sichtbar zu machen
- einige Hersteller haben sich in der Data Mining Group (DMG) zusammengeschlossen, um gemeinsam weiter am Standard zu arbeiten

5.3 Deployment und Abschluss - Modellübergabe in die Produktion

5.3.2 Modelle speichern extern mit PMML

- eine Library, welche es erlaubt, sklearn Objekte als PMML zu exportieren ist [sklearn2pmml]
- Besonderheit zum Vorgehen: die drei oben vorgestellten Modelle müssen für den Exportvorgang in einer Pipeline (vgl. Kap. 3.3.5.3) neu trainiert werden
- unten der Code zum Exportieren eines StandardScaler Objektes, mit `.fit()` als letzter aufgerufenen Methode

```
from sklearn.pipeline import Pipeline
from sklearn2pmml import PMMLPipeline, sklearn2pmml

from sklearn.preprocessing import StandardScaler
pipeline = PMMLPipeline([("scaler", StandardScaler())]).fit(X_train)
sklearn2pmml(pipeline, "StandardScaler_melb.pmml", with_repr = True)
```

- die Code Beispiele für LinearRegression und DecisionTreeRegressor finden sich im [ipynb]

5.3 Deployment und Abschluss - Modellübergabe in die Produktion

5.3.2 Modelle speichern extern mit PMML

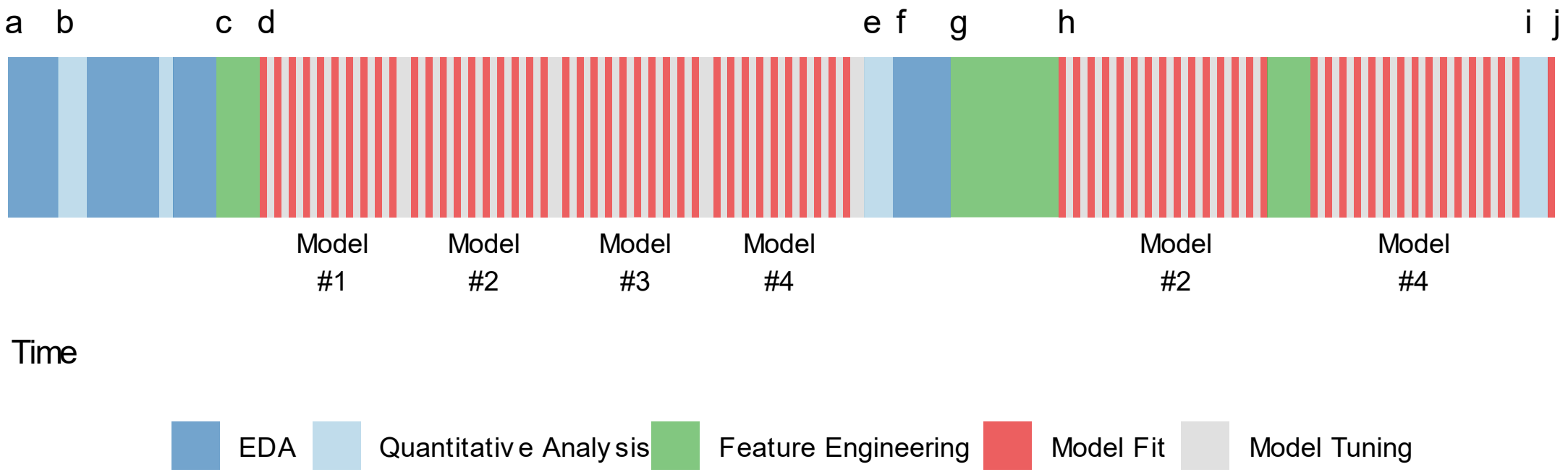
- im Gegensatz zu Dump können die so erstellten Objekte mit einem Editor inspiziert werden
- unten ein Ausschnitt aus LinearRegression_melb.pmml

```
:
        <RegressionTable intercept="-1.2823793626372582E8">
            <NumericPredictor name="Rooms" coefficient="248358.33199624726"/>
            <NumericPredictor name="Type" coefficient="-146000.56073048612"/>
            <NumericPredictor name="Distance" coefficient="-40150.92502337525"/>
            <NumericPredictor name="Bathroom" coefficient="148669.9903930434"/>
            <NumericPredictor name="Car" coefficient="41723.09686323547"/>
:
```

- gespeichert PMML-Modelle können auch wieder in Python importiert und für Predictions eingesetzt werden unter Einsatz der Library pypmml (vgl. [ipynb], raw)

5.4 Deployment und Abschluss - Der Modellierungsprozess

The Model versus the Modeling Process



5.4 Deployment und Abschluss - Der Modellierungsprozess

eine Darstellung aus Kuhn und Johnson 2019

a) erste Annäherung an die Daten mit EDA
b) erste statistische Analysen (Deskriptiv), hier auch unter Feature Exploration
c) erstes Feature Engineering aufgrund der Erkenntnisse von a) und b)
d) untersuchen und Evaluieren verschiedener Vorhersagemodelle (in der Darstellung 4) mit jeweiligem Parameter Tuning
e) vergleichen der unter d) optimierten Modelle, insb. statistische Analysen der jeweils erreichten Performance
f) basierend auf e) vertiefte EDA, insb. untersuchen des Potentials erweiterten Feature Engineerings zur Verbesserung der Performance
g) als Konsequenz aus dem obigen kann eine weitere Phase Feature Engineering angezeigt sein, ausserdem kann die Auswahl an weiter zu verfolgende Modelle eingeschränkt werden

5.4 Deployment und Abschluss - Der Modellierungsprozess

h) einschränken der weiter zu untersuchenden Modelle (hier auf zwei) durch eine weitere Phase von tunen und testen

i) finalisieren der jetzt noch betrachteten Modellkandidaten und beurteilen derselben anhand eines externen Testset

j) das final trainierte Modell wird schliesslich in Produktion gebracht

5.5 Deployment und Abschluss - PyCaret

was ist PyCaret?

- eine Open-Source, Low-Code Machine Learning Python Library zum Automatisieren von Machine Learning Workflows
- ein End-zu-End Tool für Machine Learning und Modell Management für schnellere Modellierungszyklen
- gemäss Herstellerin sind dank Low-Code Konzept für den ganzen Machine Learning Workflow nur wenige Zeilen Code notwendig, was dem Umgang vereinfachen soll
- im Wesentlichen ein Python Wrapper um verschiedene ML Libraries wie scikit-learn, XGBoost, LightGBM, CatBoost, spaCy, Optuna, Hyperopt, Ray, und viele andere
- inspiriert von caret (Classification and Regression Training), einer Library von R welche ein Framework für die verschiedenen Schritte von der Datenaufbereitung über Modellselektion und -Tuning bis Deployment unterstützt

Life Demo

(vgl. 5.5 Deployment und Abschluss - pycaret.ipynb)

5.5 Deployment und Abschluss - PyCaret

Fazit

- Low-Code trifft tatsächlich zu, allerdings weist z.B. `pycaret.classification.setup()` über 70 (!) Parameter auf, über welche das Verhalten vom Preprozessing bis zu Validierung gesteuert werden kann - so einfach ist es also dann doch nicht
- wenn man mit den Interna der verwendeten Trainingsmethoden nicht ansatzweise vertraut ist, bildet das Ganze eine riesige Black-Box!

Empfehlung

- PyCaret scheint ganz gut geeignet, um in der Anfangsphase eines Machine Learning Projektes aussichtsreiche Modelle und deren Parametrisierungen zu identifizieren
- für die Feinarbeit ist dagegen die in diesem Kurs vorgestellte "traditionelle" Vorgehensweise vorzuziehen


GESCHAFFT!
VIELEN DANK FÜR EURE
AUFMERKSAMKEIT