**Erklärung des Python-Codes für das Kundenmodell mit Spark MLlib**

Dieser Python-Code zeigt einen kompletten Workflow für ein Machine-Learning-Projekt mit Spark. Dabei durchläuft man typische Phasen wie ETL (Extract, Transform, Load), EDA (Exploratory Data Analysis) und ML (Machine Learning). Im Folgenden erkläre ich die Struktur des Codes und wie diese Prozesse umgesetzt werden.

**1. ETL-Prozess: Datenextraktion, -transformation und -laden**

Der Code beginnt mit dem ETL-Prozess, der eine wichtige Grundlage für ML-Projekte ist.

Extract (Datenextraktion):

In vorbereite\_daten() werden Rohdaten aus CSV-Dateien mit Kundendaten und künstlich generierten BMI-Werten geladen. Diese Dateien werden im nächsten Schritt in Spark eingelesen (extract\_and\_merge\_data()).

Transform (Datenumwandlung):

Die Transformation erfolgt in transform\_data(). Hier werden fehlende Werte entfernt, kategoriale Variablen (wie „Geschlecht“) in numerische Werte umgewandelt (Label-Encoding), und Features in einen Vektor zusammengefasst, der vom ML-Modell verarbeitet werden kann. Zusätzlich werden die Features skaliert (scale\_features()), um sie vergleichbar zu machen.

Load (Datenladen):

Die vorbereiteten Daten werden als Spark DataFrame für das Training und die Modellbewertung bereitgestellt. Außerdem werden die Trainings- und Testdaten mit prepare\_train\_test\_split() getrennt.

**2. EDA – Explorative Datenanalyse (implizit im Code)**

Explorative Datenanalyse (EDA) ist der Schritt, in dem man die Daten versteht, Muster erkennt und aufbereitet. Im Beispiel ist EDA nicht als separate Funktion sichtbar, aber einige Elemente des EDA-Prozesses sind enthalten:

Datenprüfung:

Fehlende Werte werden entfernt, um Probleme im ML-Prozess zu vermeiden.

Feature Engineering:

Die Umwandlung von Kategorien in Zahlen und das Zusammenfassen von Merkmalen in Vektoren sind wichtige EDA-Schritte, um die Daten in ein ML-kompatibles Format zu bringen.

Visualisierung:

Die Funktion visualize\_metrics() visualisiert die Performance der Modelle, was ein Bestandteil der EDA im weiteren Sinne ist, nämlich die Auswertung der Modellergebnisse.

**3. ML – Machine Learning: Modelle trainieren und bewerten**

Der Kern des Codes liegt im ML-Prozess:

Modellerstellung:

In build\_models() werden mehrere ML-Modelle definiert (Logistische Regression, Decision Tree mit Gini und Entropie, Random Forest). Jedes Modell bekommt einen eigenen Pipeline- und Parameterraster für Cross-Validation.

Training und Validierung:

Mit Cross-Validation werden optimale Hyperparameter für die Modelle gesucht. Das Training findet auf dem Trainingsdatensatz statt.

Evaluation:

Die Funktion evaluate\_model() berechnet wichtige Metriken (Precision, Recall, F1-Score), um die Qualität der Modelle auf den Testdaten zu bewerten.

Visualisierung und Reporting:

Die Ergebnisse werden graphisch dargestellt (visualize\_metrics()) und zusätzlich in einer Textdatei gespeichert (save\_metrics()), was hilft, die Ergebnisse nachvollziehbar zu dokumentieren.

**4. Warum sind ETL, EDA und ML so wichtig?**

Diese drei Phasen sind essenziell für jedes datengetriebene Projekt:

Ohne ETL kann man keine sauberen und gut strukturierten Daten für ML bereitstellen. Fehlerhafte oder unvollständige Daten führen zu schlechten Modellen.

EDA hilft, die Daten zu verstehen, die richtigen Features auszuwählen und potenzielle Probleme zu entdecken.

ML nutzt dann diese aufbereiteten Daten, um Modelle zu trainieren, Vorhersagen zu machen und Entscheidungen zu unterstützen.

**5. Aufbau des Codes und seine Funktionen**

Der Code besteht aus mehreren Funktionen, die jeweils einen bestimmten Teil des gesamten Ablaufs übernehmen:

**create\_spark\_session()**

Diese Funktion startet eine Spark-Session. Eine Spark-Session ist die zentrale Schnittstelle, um mit Apache Spark zu arbeiten. Sie ermöglicht das Einlesen von Daten, deren Verarbeitung und das Trainieren von Modellen. Die Funktion gibt das Spark-Session-Objekt zurück, das wir später in anderen Funktionen verwenden.

**vorbereite\_daten()**

Hier werden die ursprünglichen Kundendaten vorbereitet. Die Kundendaten liegen als CSV-Datei vor. Die Funktion lädt diese Datei mit pandas, fügt eine eindeutige ID-Spalte hinzu, die jeden Kunden identifiziert, und speichert die Daten in einer neuen Datei. Zusätzlich erzeugt sie für jeden Kunden einen zufälligen BMI-Wert (Body-Mass-Index) und speichert diese ebenfalls in einer separaten CSV-Datei. Das Ziel ist es, zwei Dateien zu haben: eine mit den Kunden-Features und eine mit den BMI-Daten, die später zusammengeführt werden.

**extract\_and\_merge\_data()**

Diese Funktion liest die beiden CSV-Dateien mit Spark ein: die Kunden-Features und die BMI-Daten. Wichtig ist hier, dass beide Dateien die Spalte „id“ besitzen. Dann werden die beiden DataFrames anhand dieser „id“ zusammengeführt (gejoint). Das Ergebnis ist ein Spark DataFrame, das alle relevanten Informationen in einer Tabelle enthält.

**transform\_data()**

Diese Funktion bereitet die Daten für das Machine Learning vor. Zuerst werden fehlende Werte entfernt, da ML-Algorithmen meistens keine Nullwerte verarbeiten können. Dann wird die Kategorie „Geschlecht“ in eine Zahl umgewandelt – dies nennt man Indexierung, um die Daten maschinenlesbar zu machen. Anschließend werden die ausgewählten Merkmale (Features) in einen einzigen Vektor zusammengefasst, der von den ML-Modellen verarbeitet werden kann. Wichtig ist, dass je nach Einstellung die BMI-Spalte mit einbezogen oder ausgeschlossen wird.

**scale\_features()**

Nach der Zusammenfassung der Features werden diese standardisiert, das heißt auf einen Mittelwert von 0 und eine Standardabweichung von 1 gebracht. Diese Skalierung ist wichtig, weil manche Modelle besser funktionieren, wenn die Daten vergleichbar skaliert sind.

**prepare\_train\_test\_split()**

Hier wird das vorbereitete DataFrame in Trainings- und Testdaten aufgeteilt. Die Trainingsdaten dienen dem Modell zum Lernen, während die Testdaten später zur Bewertung der Modellqualität genutzt werden. In unserem Fall wird 70 % der Daten zum Trainieren und 30 % zum Testen verwendet.

**build\_models()**

Diese Funktion erstellt verschiedene Machine-Learning-Modelle mit ihren Pipelines und Parameterrastern. Die Modelle sind:

* Logistic Regression
* Decision Tree mit Gini-Impurity
* Decision Tree mit Entropie
* Random Forest

Für jedes Modell wird ein Parameterraster definiert, das bei der Cross-Validation dazu dient, die besten Parameter zu finden.

**evaluate\_model()**

Diese Funktion berechnet wichtige Kennzahlen zur Bewertung der Modelle: Precision, Recall und F1-Score. Diese Metriken zeigen, wie gut das Modell Vorhersagen trifft, speziell im Kontext von Klassifikationsproblemen.

**visualize\_metrics()**

Hier werden die Bewertungen als Balkendiagramm dargestellt. Die drei Metriken werden für alle Modelle visualisiert, sodass man die Leistung auf einen Blick vergleichen kann.

**save\_metrics()**

Diese neue Funktion speichert die Metriken in einer Textdatei, damit man die Ergebnisse auch später nachlesen kann.

**main()**

Dies ist die zentrale Funktion, die alle anderen Funktionen in der richtigen Reihenfolge aufruft und so den kompletten Ablauf steuert: von der Datenvorbereitung über die Modellierung bis zur Auswertung.

**Zusammenfassung**

Der Code zeigt einen vollständigen ETL-Workflow, bei dem Daten extrahiert, transformiert und geladen werden. Dabei werden wichtige EDA-Schritte umgesetzt, um die Daten für das ML vorzubereiten. Anschließend werden mehrere ML-Modelle trainiert, validiert und bewertet.

Die klare Struktur des Codes erleichtert es, einzelne Phasen zu verstehen und bei Bedarf anzupassen oder zu erweitern. Die Visualisierung und Speicherung der Ergebnisse machen die Analyse transparent und wiederholbar.

Dieses Beispiel ist ein guter Einstieg, um den gesamten Prozess von Daten bis zur Modellbewertung in der Praxis zu erleben.