**Code-Konzept zur Vorhersage der Kreditwürdigkeit**

1. Einleitung

1.1 Ziel des Projekts

Das Ziel dieses Projekts ist die Entwicklung einer Machine Learning - Pipeline zur Vorhersage der Kreditwürdigkeit von Kunden. Dies ermöglicht es Banken, fundierte Entscheidungen bei der Kreditvergabe zu treffen, indem das Ausfallrisiko eines Kredits basierend auf historischen Daten bewertet wird.

1.2 Relevanz des Use-Cases

Die genaue Vorhersage der Kreditwürdigkeit ist für Finanzinstitute von entscheidender Bedeutung, um Verluste zu minimieren und profitable Geschäftsbeziehungen zu fördern. Ein datengetriebenes Modell bietet eine objektive Grundlage für Kreditentscheidungen.

2. Klassenbeschreibung

2.1 Customer

class Customer:

def \_\_init\_\_(self, name, age, income, loan\_amount, number\_of\_loans):

self.name = name

self.age = age

self.income = income

self.loan\_amount = loan\_amount

self.number\_of\_loans = number\_of\_loans

- Attribute:

- `name` (str): Name des Kunden

- `age` (int): Alter des Kunden

- `income` (float): Einkommen des Kunden

- `loan\_amount` (float): Beantragte Kreditsumme

- `number\_of\_loans` (int): Anzahl der laufenden Kredite

2.2 CreditScorer

from sklearn.ensemble import RandomForestClassifier

class CreditScorer:

def \_\_init\_\_(self):

self.model = RandomForestClassifier()

def train\_model(self, data):

X, y = data.drop('target', axis=1), data['target']

self.model.fit(X, y)

def predict(self, customer):

features = [[customer.age, customer.income, customer.loan\_amount, customer.number\_of\_loans]]

return self.model.predict(features)

- Attribute:

- `model` (RandomForestClassifier): Das Machine Learning Modell zur Vorhersage

- Methoden:

- `train\_model(data)`: Trainiert das Modell mit den bereitgestellten Daten

- `predict(customer)`: Gibt eine Vorhersage für die Kreditwürdigkeit des Kunden

2.3 DataLoader

import pandas as pd

class DataLoader:

def \_\_init\_\_(self, data\_source):

self.data\_source = data\_source

def load\_data(self):

return pd.read\_csv(self.data\_source)

def preprocess(self, data):

# Beispielhafte Vorverarbeitung

data.fillna(data.mean(), inplace=True)

return data

- Attribute:

- `data\_source` (str): Pfad zur Datenquelle

- Methoden:

- `load\_data()`: Lädt die Daten aus der angegebenen Quelle

- `preprocess(data)`: Führt Datenbereinigung und -transformation durch

2.4 Evaluation

from sklearn.metrics import accuracy\_score

class Evaluation:

def evaluate\_model(self, model, test\_data):

X\_test, y\_test = test\_data.drop('target', axis=1), test\_data['target']

predictions = model.predict(X\_test)

return accuracy\_score(y\_test, predictions)

- Methoden:

- `evaluate\_model(model, test\_data)`: Bewertet die Genauigkeit des Modells

3. Interaktion der Objekte

3.1 Sequenzdiagramm

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

4. Coding-Guidelines

4.1 PEP 8 Konformität

- Der Code folgt den PEP 8 Richtlinien, um Lesbarkeit und Konsistenz zu gewährleisten.

- Beispielsweise wird für die Benennung von Variablen und Funktionen das `snake\_case` verwendet.

4.2 Kommentare und Dokumentation

- Jeder wichtige Abschnitt des Codes wird mit Kommentaren versehen, um die Funktionsweise zu erläutern.

- Funktionen und Klassen enthalten Docstrings, die ihre Funktion beschreiben.

4.3 Modularität und Wiederverwendbarkeit

- Der Code ist in modularen Komponenten organisiert, die wiederverwendbar und erweiterbar sind.

4.4 Logging

- Anstelle von Print-Statements wird das Logging-Modul verwendet, um Informationen und Fehler zu protokollieren.

5. Aufwandsschätzung und Effiziente Implementierung

5.1 Vorbereitungsphase (ca. 20 Stunden)

- Datensammlung, Anforderungsanalyse, Design der Klassenstruktur

5.2 Implementierungsphase (ca. 40 Stunden)

- Entwicklung der einzelnen Module und Integration der Pipeline

5.3 Testphase (ca. 20 Stunden)

- Erstellen von Unit-Tests und Durchführung von Integrationstests

5.4 Optimierungsphase (ca. 10 Stunden)

- Performance-Tuning und Code-Refactoring

5.5 Dokumentations- und Präsentationsphase (ca. 10 Stunden)

- Erstellung der Dokumentation und Vorbereitung der Präsentation

6. Zusammenfassung und Ausblick

Dieses Projekt zielt darauf ab, eine Machine Learning - Pipeline zur Vorhersage der Kreditwürdigkeit zu entwickeln. Es werden verschiedene Klassen für die Datenverarbeitung, Modelltraining und -bewertung verwendet. Durch die Einhaltung von Coding-Guidelines und einer klaren Struktur wird eine effiziente Implementierung und Wartung des Codes gewährleistet. Zukünftige Erweiterungen könnten zusätzliche Datenquellen und fortschrittlichere Modelle umfassen.