Projekt „Give me loan”

Give **me** loan!

The entire loan system support pipeline

projekt ASI
wersja 1.0

Autorzy projektu:
Adrian Kordas (s19567)
Piotr Skibiński (s19369)
Wojciech Szczepański (s18345)

Contents

DZIEDZINA PROBLEMOWA 3
CEL 3
PRZEZNACZENIE DOKUMENTU 4
ZBIÓR DANYCH 4
TECHNOLOGIA 5
ZADANIA APLIKACJI 8
Czytanie parametrów, kompaktowość 8
Zapisywanie logów 9
Trenowanie 9
Predykcja 11
Jakość systemu i wnioski 11

DZIEDZINA PROBLEMOWA

Pożyczki są szybkim sposobem na uzyskanie dodatkowej ilości gotówki. Są niezwykle popularnym produktem finansowym, po który sięgają osoby jej potrzebujące dla różnych celów. Według definicji pożyczka gotówkowa jest umową, na podstawie której pożyczkodawca udostępnia środki pieniężne pożyczkobiorcy. Umowa określa termin zwrotu pieniędzy w takiej samej ilości oraz dodatkowych opłat z nią związanych. Pożyczek mogą udzielać banki lub firmy pozabankowe. W celu uzyskania pożyczki osoba musi spełniać szereg warunków przedstawionych przez instytut udzielający kredytu. Bardzo często zdarza się, że osoby potrzebujące gotówki nie otrzymują wsparcia finansowego z powodu niespełniania warunków tj:

1) Zarobki pożyczkobiorcy nie pozwalają na wzięcie kredytu
2) Pożyczkobiorca nie spłaca swoich zobowiązań
3) Warunki zatrudnienia nie spełniają wymaganych kryteriów
4) Wiek nie spełnia warunków (osoby młode lub w podeszłym wieku)

Zdarza się tak, że osoba potrzebująca gotówki traci czas na składanie zapytań do firm udzielających pożyczki licząc na to, że otrzyma wsparcie. Jednakże nie ma pojęcia jaką decyzję podejmie firma lub bank. Z drugiej strony instytucje udzielające kredytu nie są pewne, czy klient będzie w stanie spłacać pożyczkę w terminie. W celu umożliwienia szybkiej weryfikacji uzyskania kredytu została stworzona aplikacja, która szacuje czy klient ma szansę uzyskać pożyczkę.

CEL

Głównym celem projektu jest wsparcie klientów w oszacowaniu możliwości wzięcia pożyczki lub pracowników banku w podejmowaniu decyzji o udzieleniu kredytu. Zastosowanie metod uczenia maszynowego może znacznie skrócić proces wydawania decyzji kredytowej. Projekt obejmuje cały potok, w tym szkolenie, ponowne szkolenie, tworzenie nowego modelu i wykonywanie prognoz. Aplikacja umożliwia wytrenowanie najlepszego algorytmu uczenia maszynowego dostępnego w bibliotece ***pycaret*** oraz nadzór nad aplikacją. Nadzór realizowany jest za pomocą komponentu ***mlflow*** oraz wbudowanego systemu logów, który prezentuje informacje o wykrytym dryfie, wykryciu nowego modelu itp.

PRZEZNACZENIE DOKUMENTU

Dokument stanowi opis przygotowanego rozwiązania. Pełni ważną rolę w zrozumieniu poruszonego zagadnienia wraz z przejściem przez aspekt techniczny.

ZBIÓR DANYCH

Zbiór danych pozyskany został ze strony internetowej **kaggle.com**. Zawiera on istotne dane klientów w kontekście przyznawania pożyczki. Dane, które przechowuje zbiór to:

1. ID wniosku
2. Płeć
3. Stan matrymonialny
4. Czy wnioskujący ma jakąś osobę na utrzymaniu
5. Stopień wykształcenia
6. Czy osoba jest na samozatrudnieniu
7. Przychód
8. Przychód partnera / partnerki
9. Wysokość pożyczki
10. Długość spłaty
11. Czy wnioskodawca posiada historię kredytową
12. Obszar zabudowania wnioskodawcy
13. Decyzja o zaakceptowaniu wniosku

Dane zostały wyczyszczone i uporządkowane. W tym celu wykorzystane zostały funkcje oferowane przez pakiet ***Pandas***. Działania, które zastosowano wobec zbioru danych:

- **Usunięcie zbędnych kolumn**

Na początku przygotowania danych wyrzucane są kolumny, które nie wnoszą żadnej wartości w kontekście uczenia modelu. Wobec tego odrzucana zostaje kolumna ID Wniosku (Loan_ID) za pomocą funkcji ***drop(kolumna, axis=1).*** Parametr ***axis=1*** oznacza, że funkcja ma usunąć całą kolumnę z danymi.

- **Grupowanie danych**

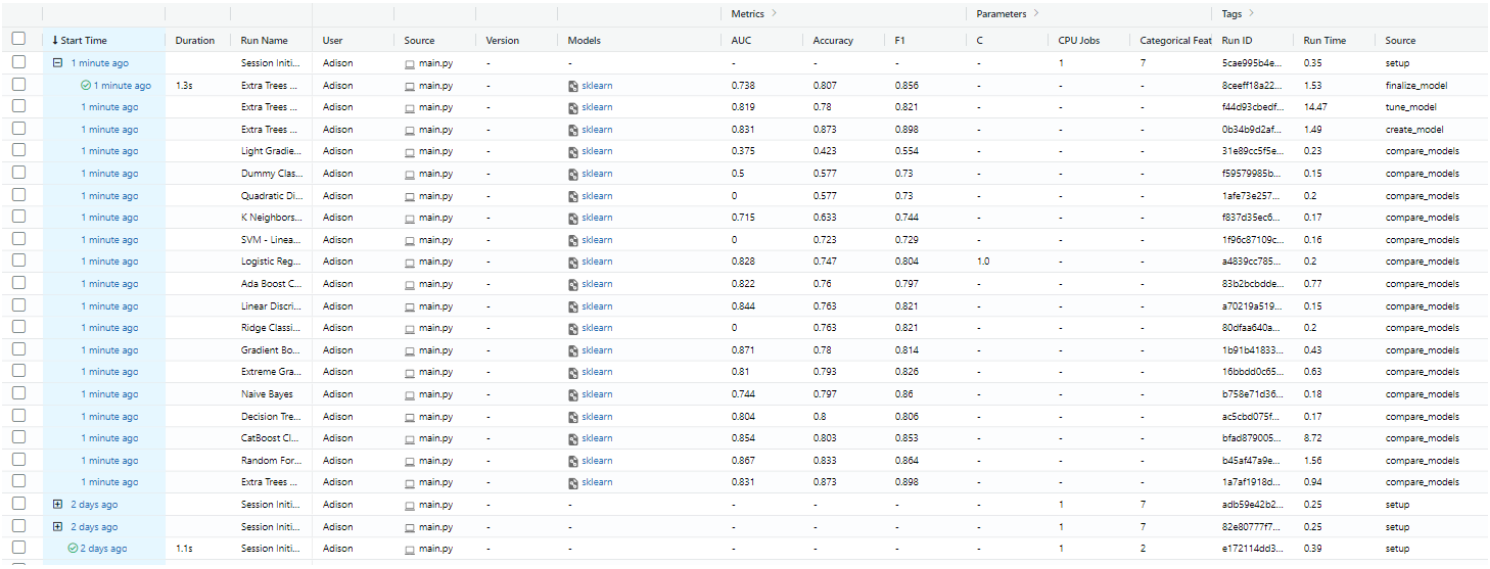
Wiele kolumn zawiera dane grupujące wnioskodawców. Na przykład płeć (opisywana tekstowo – Male lub Female), stopień wykształcenia (Graduate lub Not Graduate). Aby usprawnić proces uczenia modelu zastosowano funkcje grupujące na rzecz konkretnych kolumn. Odpowiednie grupy zostały oznaczone liczbami. Wobec tego w zbiorze danym wejściowym do funkcji trenującej płeć traktowana jest jako 0 lub 1. Podobnie z pozostałymi kolumnami klasyfikującymi. Do wykonania tego zadania zastosowana została funkcja ***df.groupby(kolumna).ngroup()***

- **Usuwanie rekordów zawierających puste komórki**

Żadna z informacji, która jest zawarta we wniosku nie nadaje się do uśrednienia (jeżeli jej brakuje). Dane jak przychód, wysokość pożyczki itp. nie powinny być uśredniane. W związku z tym na etapie projektowym podjęto decyzję, aby odrzucać wszystkie rekordy ze zbioru danych, w których komórki zawierają puste wartości. Do wykonania tego zadania użyta została funkcja ***dropna(inplace=True).***

TECHNOLOGIA

Do trenowania modelu wykorzystana została biblioteka ***pycaret***. Pozwala ona na szybkie uczenie z możliwie najmniejszą ilością kodu. Dodatkowo daje możliwość wygodnego porównania jakości wielu algorytmów uczenia maszynowego w kontekście tego samego zbioru danych. Dodatkowo użyta została biblioteka ***mlflow***, która śledzi działania wykonywane w aplikacji.



							Metrics			Parameters			Tags		
Start Time	Duration	Run Name	User	Source	Version	Models	AUC	Accuracy	F1	C	CPU Jobs	Categorical Feat	Run ID	Run Time	Source
1 minute ago		Session Initi...	Adison	main.py	-	-	-	-	-	-	1	7	5cae995b4e...	0.35	setup
1 minute ago	1.3s	Extra Trees ...	Adison	main.py	-	sklearn	0.738	0.807	0.856	-	-	-	8ceeff18a22...	1.53	finalize_model
1 minute ago		Extra Trees ...	Adison	main.py	-	sklearn	0.819	0.78	0.821	-	-	-	f44d93cbedf...	14.47	tune_model
1 minute ago		Extra Trees ...	Adison	main.py	-	sklearn	0.831	0.873	0.898	-	-	-	0b34b9d2af...	1.49	create_model
1 minute ago		Light Gradie...	Adison	main.py	-	sklearn	0.375	0.423	0.554	-	-	-	31e89cc5f5e...	0.23	compare_models
1 minute ago		Dummy Clas...	Adison	main.py	-	sklearn	0.5	0.577	0.73	-	-	-	f59579985b...	0.15	compare_models
1 minute ago		Quadratic Di...	Adison	main.py	-	sklearn	0	0.577	0.73	-	-	-	1afe73e257...	0.2	compare_models
1 minute ago		K Neighbors...	Adison	main.py	-	sklearn	0.715	0.633	0.744	-	-	-	f837d35ec6...	0.17	compare_models
1 minute ago		SVM - Linea...	Adison	main.py	-	sklearn	0	0.723	0.729	-	-	-	1f96c87109c...	0.16	compare_models
1 minute ago		Logistic Reg...	Adison	main.py	-	sklearn	0.828	0.747	0.804	1.0	-	-	a4839cc785...	0.2	compare_models
1 minute ago		Ada Boost C...	Adison	main.py	-	sklearn	0.822	0.76	0.797	-	-	-	83b2bcbdde...	0.77	compare_models
1 minute ago		Linear Discri...	Adison	main.py	-	sklearn	0.844	0.763	0.821	-	-	-	a70219a519...	0.15	compare_models
1 minute ago		Ridge Classi...	Adison	main.py	-	sklearn	0	0.763	0.821	-	-	-	80dfaa640a...	0.2	compare_models
1 minute ago		Gradient Bo...	Adison	main.py	-	sklearn	0.871	0.78	0.814	-	-	-	1b91b41833...	0.43	compare_models
1 minute ago		Extreme Gra...	Adison	main.py	-	sklearn	0.81	0.793	0.826	-	-	-	16bbdd0c65...	0.63	compare_models
1 minute ago		Naive Bayes	Adison	main.py	-	sklearn	0.744	0.797	0.86	-	-	-	b758e71d36...	0.18	compare_models
1 minute ago		Decision Tre...	Adison	main.py	-	sklearn	0.804	0.8	0.806	-	-	-	ac5cbd075f...	0.17	compare_models
1 minute ago		CatBoost Cl...	Adison	main.py	-	sklearn	0.854	0.803	0.853	-	-	-	bfad879005...	8.72	compare_models
1 minute ago		Random For...	Adison	main.py	-	sklearn	0.867	0.833	0.864	-	-	-	b45af47a9e...	1.56	compare_models
1 minute ago		Extra Trees ...	Adison	main.py	-	sklearn	0.831	0.873	0.898	-	-	-	1a7af1918d...	0.94	compare_models
2 days ago		Session Initi...	Adison	main.py	-	-	-	-	-	-	1	7	adb59e42b2...	0.25	setup
2 days ago		Session Initi...	Adison	main.py	-	-	-	-	-	-	1	7	82e80777f7...	0.25	setup
2 days ago	1.1s	Session Initi...	Adison	main.py	-	-	-	-	-	-	1	2	e172114dd3...	0.39	setup

Rys. 1. Wyniki uczenia przedstawione w mlflow.

Dane ***mlflow*** prezentowane są na serwerze http na podstawie danych zapisywanych w projekcie w katalogu ***mlruns***. Aby uruchomić serwer http ***mlflow*** będąc w katalogu projektu należy użyć komendy:

mlflow ui

Aplikacja może zostać uruchomiona na dwa sposoby. Za pomocą klasycznego wiersza poleceń lub za pomocą obrazu ***docker***. Stworzenie obrazu dockera (aplikacji do konteneryzacji projektów) wymagało stworzenia dodatkowego pliku – ***dockerfile***. Plik ten zawiera informacje o projekcie, przygotowaniu do uruchomienia oraz jak aplikacja powinna być traktowana w obrazie.

Polecenia w pliku ***dockerfile*** wyglądają następująco:

```
FROM python:3.8-slim-buster

WORKDIR /app

RUN apt-get update && apt-get install -y --no-install-recommends apt-utils
RUN apt-get -y install curl
RUN apt-get install libgomp1

RUN pip install pycaret
RUN pip install numpy
RUN pip install pandas
RUN pip install plotly
RUN pip install mlflow

COPY /app .
COPY /app/train.py train.py


ENTRYPOINT ["python", "main.py"]
```

Rys2. Zawartość pliku dockerfile

Na początku wskazany został interpreter kodu. Następnie wskazano katalog główny aplikacji, zainstalowane zostały niezbędne biblioteki (***docker*** działa na jądrze unix). Po wyposażeniu środowiska w niezbędne biblioteki skopiowana została zawartość katalogu z projektem do obrazu ***docker***. Ostatnia linia pliku ***dockerfile*** zawiera informację, że start aplikacji rozpoczyna się od pliku main.py i możemy przesyłać do niej różne parametry.

Kontener ***docker*** został przygotowany przy pomocy komendy:

docker build -t python-image .

I można nim zarządzać również z poziomu Docker-GUI

Rys3. Kontener widoczny w Docker GUI

Kontener można uruchomić na dwa sposoby:

docker run –publish 8000:8000 python-image

docker run python-image

Dodatkowo do poprawnego działania aplikacji do polecenia należy dodać parametry wysyłane do pliku main.py. Parametry zostaną przedstawione w kolejnej części dokumentacji.

ZADANIA APLIKACJI

Czytanie parametrów, kompaktowość

Aplikacja powinna być uruchamiana z jednego miejsca, łatwo i prosto. W tym celu przygotowany został interpreter argumentów wejściowych z dodatkową opcją pomocy.

Wyświetlenie dostępnych parametrów powoduje komenda:

python main.py -h

I zwraca następujące informacje:

```
PS C:\Users\Adison\Desktop\Grupa-2-main\app> python main.py -h
usage: main.py [-h] [--goal GOAL] [--data DATA] [--drift DRIFT] [--dataset DATASET]

Parametry aplikacji

optional arguments:
  -h, --help         show this help message and exit
  --goal GOAL        Cel, jaki chcesz osiągnąć (train / predict / post-training)
  --data DATA        Ścieżka do zbioru przeznaczonego to predykcji. Format : '<ścieżka do pliku>'
  --drift DRIFT      Czy wygenerować dane do wykrycia dryfu (y/n)
  --dataset DATASET  Ścieżka do zbioru danych do treningu. Format : '<ścieżka do pliku>'
```

Rys 4. Wyświetlenie pomocy

Najważniejsze polecenia dla aplikacji to:

python main.py --goal predict --data 'data/data_test.csv' – polecenie przekazujące aplikacji informację, że użytkownik chce wykonać predykcję danych umieszczonych w katalogu data/data_test.csv.

python main.py --goal train --drift n --dataset 'data/data_train.csv' – polecenie przekazujące aplikacji informację, że użytkownik chce wytrenować nowy model, a zbiór danych znajduje się w pliku data/data_train.csv. Przełącznik --drift może przekazywać dwie informacje – y lub n (tak lub nie). Jest to wyzwalacz, który dodaje wcześniej spreparowane błędne dane do zbioru, aby wywołać detekcję dryfu danych.

python main.py --goal post-training –dataset 'data/additional.csv' – polecenie przekazuje informację, że użytkownik chce dotrenować istniejący model. Wówczas do obecnego zbioru danych dodawana jest dodatkowa porcja informacji pochodząca z pliku data/additional.csv.

Zapisywanie logów

Aplikacja ma możliwość zapisywania logów po trenowaniu, dotrenowaniu, czy też przy detekcji dryfu. Pozwala to na łatwe zarządzanie aplikacją i sprawdzanie jej aktualnego stanu, oraz historii.

```
[2022-06-17 00:00:00] FIRST MODEL ARRIVED! ACCURACY: 939
[2022-06-17 00:00:00] DRIFT DETECTED. OLD ACCURACY : 939 NEW ACCURACY : 938
[2022-06-17 00:00:00] NEW MODEL IN THE FAMILY! ACCURACY : 939. OLD MODEL "model_1_939_acc.pkl" IS IN "MODELS/OLD"
[2022-06-17 00:00:00] NEW MODEL IN THE FAMILY! ACCURACY : 939. OLD MODEL "model_2_939_acc.pkl" IS IN "MODELS/OLD"
[2022-06-17 00:00:00] NEW MODEL IN THE FAMILY! ACCURACY : 939. OLD MODEL "model_3_939_acc.pkl" IS IN "MODELS/OLD"
[2022-06-17 00:00:00] DRIFT DETECTED. OLD ACCURACY : 939 NEW ACCURACY : 934
```

Rys. 5. Fragment pliku logów

Aplikacja zapisuje logi, gdy pojawi się nowy model (wcześniej nie było żadnego), gdy nowy model jest lepszy od poprzedniego i gdy wykryty został dryf danych.

Trenowanie

Trenowanie jest kluczowym elementem całej infrastruktury projektu. Wszystkie działania odbywają się w pliku ***train.py***. Najpierw dane podane przez użytkownika są obrabiane (opisane w punkcie pt. Zbiór danych). W strukturze plików projektu zostały stworzone trzy katalogi. Pierwszy z nich to katalog o nazwie ***dataset***, drugi ***models*** i trzeci ***old***. Ostatni znajduje się w katalogu nadrzędnym ***models***.

Niniejszy rozdział zostanie podzielony na trzy punkty: tworzenie całkiem nowego modelu, tworzenie modelu i porównanie go z istniejącym, dotrenowanie.

- **Tworzenie nowego modelu**

Podczas tworzenia nowego modelu – katalog ***dataset*** i ***models*** jest pusty. Po takiej weryfikacji przez aplikację – zbiór danych jest zapisywany do katalogu ***dataset***. Jeżeli na wejściu został podany argument wywołujący dryf danych – wówczas do zbioru w pamięci podręcznej aplikacji zostaje dodana dodatkowa porcja danych mająca na celu zmniejszyć ostateczną precyzję modelu.

Po przygotowaniu danych przygotowywany zostaje ***pipeline*** projektu. Odbywa się to przy pomocy funkcji ***setup*** dostarczanej przez bibliotekę ***pycaret***. Do funkcji przekazywane są parametry takie jak zbiór danych (typu ***dataframe*** biblioteki ***pandas***), cel klasyfikacji (która kolumna będzie podlegała klasyfikacji), nazwa eksperymentu, oraz dodatkowe parametry uzupełniające, jak np. ***silent=True*** wskazujący, że aplikacja ma działać bez ingerencji administratora.

Następnie uruchamiana jest funkcja ***compare_models***, która zwraca do zmiennej ***best algorytm***, które ma najlepsze predyspozycje do wyuczenia wskazanego zbioru. Po tym wykonywane są funkcje optymalizujące model i jego jakość (współczynnik ***accuracy***).

Aplikacja zapisuje informację o tym, że w systemie pojawił się nowy model i zapisuje go w formacie .pkl w katalogu ***models***. Plik zawiera w nazwie współczynnik ***accuracy*** oraz liczbę porządkową.

- **Trenowanie modelu i porównanie go z istniejącym**

Gdy w katalogu ***models*** istnieje już jakiś model aplikacja nie podmienia od razu zbioru danych, ani modelu. Proces trenowania jest taki sam, jak w punkcie powyżej, lecz dodatkowo stawiane jest pytanie:

Czy obecny model jest lepszy, niż poprzedni?

Jeżeli tak – należy stary model przenieść do katalogu models/old, a nowy zapisać w katalogu ***models*** z kolejną liczbą porządkową i nową wartością ***accuracy***. Stary zbiór danych zostaje usunięty i zastąpiony nowym. System logów zapisuje stosowną informację.

Jeżeli nie – oznacza to, że wykryto dryf i zmiany zostają odrzucone. System logów zapisuje informację o wykrytym dryfie.

- **Dotrenowywanie**

Plik z danymi do dotrenowania jest w takim samym kształcie jak plik zapisany w katalogu ***dataset***. Do istniejącego zbioru danych dodawana jest paczka dotrenowywania przy pomocy funkcji ***concat*** wchodzącej w skład biblioteki ***pandas***. Sposób trenowania jest taki sam, jak w poprzednich punktach. Na koniec porównywana jest jakość modelu tak samo, jak w przypadku trenowania przy istniejącym modelu. Jeżeli okaże się, że jakość modelu zwiększyła się to wówczas nowy zbiór danych (z dodatkowymi rekordami pochodzącymi z pliku dotrenowania) oraz model zostają zapisane w katalogach docelowych.

Predykcja

System oferuje funkcję predykcji danych. Na wejściu podawana jest ścieżka do zbioru danych, które należy sklasyfikować. Predykcja odbywa się w pliku ***predict.py*** , przygotowuje dane przy pomocy tych samych funkcji co podczas trenowania, podłącza się do ***pipeline*** za pomocą funkcji ***setup***. Odczytywany zostaje model w formacie ***.pkl*** znajdujący się w katalogu ***models***, a następnie przy pomocy funkcji ***predict_model*** wchodzącej w skład biblioteki ***pycaret*** wykonywana jest predykcja. Wynik wyświetlany jest na konsoli.

	Gender	Married	Dependents	Education	Self_Employed	...	Credit_History	Property_Area	Loan_Status	Label	Score
1	1	1	1	0	0	...	1.0	0	0	0	1.0000
2	1	1	0	0	1	...	1.0	2	1	1	0.9333
3	1	1	0	1	0	...	1.0	2	1	1	0.8155
4	1	0	0	0	0	...	1.0	2	1	1	0.8303
5	1	1	2	0	1	...	1.0	2	1	1	0.5044
6	1	1	0	1	0	...	1.0	2	1	1	0.8155
7	1	1	3	0	0	...	0.0	1	0	0	1.0000
8	1	1	2	0	0	...	1.0	2	1	1	0.5044

Rys. 6. Przykładowy wynik predykcji

Jakość systemu i wnioski

System działa dobrze, radzi sobie z dużymi zbiorami danych. Przygotowany jest do klasyfikacji wieloklasowej, lub binarnej. W przypadku niniejszego projektu jest to klasyfikacja binarna. Biblioteka ***pandas*** czasami może sprawiać problemy przy wykonywaniu predykcji błędnie obliczając ilość wartości (kolumn) na wejściu. Wówczas należy uruchomić predykcję jeszcze raz.

Aplikacja jest przygotowana na wiele błędów – jak na przykład odmowa wykonania predykcji, gdy w pliku ***models*** nie znajduje się plik modelu itp.

Dzięki systemowi interpretacji argumentów aplikacja jest bardzo kompaktowa i łatwa w obsłudze. Dodatkowy katalog ***models/old***, oraz plik logów znacznie ułatwią administratorowi nadzór nad całą architekturą systemu.

Projekt został napisany w sposób pozwalający na bardzo wygodną, oraz szybką zmianę zbioru danych, oraz dodatkowych parametrów treningu. Każda istotna czynność została opakowana indywidualną funkcją w pliku *tools.py* dzięki czemu modyfikacja systemu nie powinna sprawiać żadnych problemów.

Zastosowanie obrazu ***Docker*** pozwala na wygodną przenaszalność aplikacji. Oprócz tego wygenerowany został plik ***requirements.txt***, który zawiera niezbędne biblioteki instalacyjne. Mimo tego zastosowana została minimalna ilość bibliotek z uwagi na to, iż w niektórych firmach dostęp do nich może być ograniczony.

Użycie komponentu ***mlflow-tracking*** pozwala na rozszerzoną diagnozę systemu i istniejącego w nim modelu, oraz daje możliwość administratorowi na jeszcze większą kontrolę nad całym ekosystemem środowiska.