DOKUMENTACJA

WERSJA 1.1

Jakub Wudarski, Kamil Majchrzak, Franciszek Baron, Aleksandra Skierska, Stanisław Knapiński

Spis treści

Opis Problemu

Projekt ma na celu stworzenie modelu przewidującego gatunek pingwina na podstawie określonych atrybutów. Dodatkowo udostępnianie tego modelu opiera się na technologii API, które na podstawie przesłanych danych w żądaniu HTTP będzie oceniać, jaki jest gatunek pingwina.

Cel Projektu

Celem projektu jest stworzenie oraz wybranie odpowiedniego modelu do danych prezentowanych przez badany zbiór. Przeprowadzenie optymalizacji modelu w celu usprawnienia i zmaksymalizowania jego rezultatów, a także uzyskania jakościowej predykcji charakteryzującej się wysoką skutecznością.

Projekt ma zapewnić poprawne działanie zarówno na poziomie predykcji jak i działania API, a korzystanie z rezultatu powinno być ogólnodostępne i proste w użyciu dla każdego zainteresowanego.

Model ML

Opis Zbioru danych

Badany zbiór danych to zbiór parametrów dotyczących charakterystyki różnych gatunków pingwinów z różnych wysp.

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
...	...	...	...	...	...	...	...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	FEMALE
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	MALE
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	FEMALE
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	MALE

Opis cech

Zbiór zawiera w sobie **6 cech** oraz jedną zmienną predykcyjną, którą będzie cecha **'species'**.

Poniżej przedstawiono poniższe cechy:

a) Island – wyspa pochodzenia
b) Bill_length_mm – długość dziobu
c) Bill_depth_mm – głębokość dziobu
d) Flipper_length_mm – długość płetwy
e) Body_mass_g – masa ciała podana w gramach
f) Sex – płeć

Wartości brakujące

Zbiór danych posiada 344 wiersze, z czego **333** może być brane pod uwagę, z racji na wartości brakujące w poszczególnych kolumnach

```
df.isna().sum()
```

```
species              0
island               0
bill_length_mm       2
bill_depth_mm        2
flipper_length_mm    2
body_mass_g          2
sex                 11
dtype: int64
```

Analiza zbioru i rozkład danych

a) Bill_length_mm: Średnia: 43.992793, Maks: 59.6000 , b) Min: 32.1000, Odchylenie: 5.468668
b) Bill_depth_mm: Średnia: 17.164865 , Maks: 21.5000 , b) Min: 13.1000 , Odchylenie: 1.969235
c) Flipper_length_mm: Średnia: 200.966967, Maks: 231.000000, b) Min: 172.000000, Odchylenie: 14.015765
d) Body_mass_g: Średnia: 4207.057057, Maks: 6300.000000 , Min: 2700.000000, Odchylenie: 805.215802

Zamiana wartości kategorycznych na liczbowe

Do zakodowania kategorialnych atrybutów jako liczby całkowite używany jest LabelEncoder. Funkcja ta przygotowuje dane do użycia w modelu machine learning, eliminując konieczność pracy z wartościami tekstowymi i zamieniając je na liczby całkowite. To jest częsty krok w przygotowaniu danych przed trenowaniem modelu, ponieważ wiele algorytmów machine learning wymaga, aby dane były w formie liczbowej.

Kolumny, które są poddawane kodowaniu:

a) "Island"
b) "sex”
c) "species"

Zbiory Treningowe/Testowe

Zbiór Danych zostaje podzielony na zbiór treningowy i testowy za pomocą Biblioteki Scikit-learn. Dzieli zbiór w taki sposób, że 20% procent oryginalnego zbioru zostaje umieszczone w zbiorze testowym. Takim, którego algorytm nie widzi podczas treningu, oraz 80% do zbioru treningowego.

Algorytm TabularPredictor

TabularPredictor to narzędzie z biblioteki **autogluton.tabular**, które automatyzuje proces budowy modeli dla problemów związanych z danymi tabularnymi. W aplikacji, przewidującej gatunek pingwinów na podstawie różnych cech, **TabularPredictor** samodzielnie dostosowuje różne modele i ich hiperparametry, aby znaleźć optymalne rozwiązanie. Dzięki temu nie musimy ręcznie dostosowywać modeli, co pozwala skoncentrować się na samej analizie danych i uzyskiwaniu dokładnych prognoz.

Architektoniczny model systemu

Technologie

Fast API

Flask to mikroframework webowy napisany w języku Python. Jest to minimalistyczny, lekki i bardzo elastyczny narzędzie do tworzenia aplikacji internetowych.

Flask umożliwia programistom budowanie serwerów aplikacji i API w sposób prosty i przejrzysty, przy minimalnej ilości gotowych założeń i struktur. W systemie został użyty z powodu konieczności integracji modeli sztucznej inteligencji oraz stworzonego API. Flask API zawiera 5 końcówek o takiej samej budowie przyjmujących tyle samo argumentów oraz zwracających dokładnie jeden wynik – prognozę dla jednego z pięciu modeli.

Kedro

Kedro to framework do zarządzania projektami danych w języku Python, stworzony z myślą o prostocie, modularności i elastyczności. Jest to narzędzie dedykowane projektom z obszaru analizy danych, umożliwiające efektywne zarządzanie procesami przetwarzania danych. Poniżej znajdziesz analogie do opisu Flask API, ale w kontekście Kedro.

Struktura projektu

Kedro zapewnia zorganizowaną strukturę projektu, z katalogami **conf, data, src, logs, notebook, results.**

W **src** znajdują się pipeline'y danych oraz "nodes", reprezentowane jako funkcje Pythona, realizujące konkretne zadania.

Pipeline

Definiowanie pipeline'ów danych w plikach YAML, które określają sekwencję etapów przetwarzania danych. Każdy etap to "node" (funkcja), a pipeline to sekwencja wywołań tych funkcji.

Nodes

"Nodes" to funkcje Pythona reprezentujące konkretne operacje przetwarzania danych w pipeline'ie. Każda funkcja przyjmuje określone wejście i generuje konkretne wyjście.

Docker

Na bazie Dockerfile tworzymy obrazu Docker dla całego projektu, wykorzystującego Conda i Rust. Oto szczegółowy opis jego zawartości:

Continuumio/miniconda3: Ta linia określa bazowy obraz, którym w tym przypadku jest obraz z zainstalowaną minicondą. Miniconda to minimalny instalator dla Conda, systemu zarządzania pakietami i środowiskami open-source z którego korzystamy w projekcie.

Instalacja Zależności Systemowych:

Polecenie RUN aktualizuje listy pakietów i instaluje curl oraz build-essential za pomocą apt-get. Są to podstawowe narzędzia do pobierania i kompilacji oprogramowania.

Instalacja Rust:

Instalujemy Rust, z którego korzystamy podczas instalacji pakietów.

Polecenie ENV służy do aktualizacji zmiennej środowiskowej PATH, aby zawierała ścieżkę instalacji Rust.

Przy pomocy rustup default stable ustawiamy stabliną wersję Rust jako domyślną.

Przy pomocy WORKDIR /usr/src/app ustawiamy katalog roboczy wewnątrz kontenera Docker na /usr/src/app.

Kopiujemy Pliki Projektu używając polecenie COPY . . kopiuje pliki projektu do katalogu roboczego w kontenerze.

Konfiguracja Środowiska Conda:

RUN conda env create -f environment.yml: Tworzy środowisko Conda na podstawie pliku environment.yml z projektu.

Polecenie SHELL jest skonfigurowane do używania środowiska Conda dla kolejnych poleceń RUN.

Środowisko jest aktywowane przy użyciu source activate penguins-env.

Uruchamianie pipeliny Kedro:

Plik zawiera polecenia do uruchomienia pipelinów Kedro do przetwarzania wstępnego i modelowania danych (RUN kedro run -p preprocessing i RUN kedro run -p modeling).

EXPOSE 8000: Ta instrukcja informuje Docker, że kontener nasłuchuje na porcie 8000.

Uruchamianie FastAPI z Uvicorn:

Ostateczne polecenie uruchamia FastAPI za pomocą Uvicorn, gdy kontener jest włączany. Aktywuje środowisko Conda i uruchamia aplikację FastAPI na porcie 8000.

Dokumentacja API – Swagger

Dokumentacja interfejsu REST API została stworzona za pomocą narzędzia Swagger. To narzędzie umożliwia nie tylko wizualizację, lecz także interakcję z API poprzez dynamicznie generowaną i aktualizującą się dokumentację.

Komunikacja z Systemem w zakresie pobierania danych bieżących oraz historycznych odbywa się poprzez interfejs REST API

• Dane są dostarczone w formacie danych JSON

• Interfejs API udostępnia predykcję na podstawie otrzymanych parametrów

• Format odpowiedzi przygotowany jest w formacie JSON

• Obsługa błędnych zapytań jest obsłużona komunikatem o przyczynie, kody błędów lub poprawnych odpowiedzi zgodne z kodami odpowiedzi http

Zasoby i Endpoint

POST /run-pipeline

Opis: Endpoint zwraca predykcję gatunku pingwina, dla podanych parametrów.

Wymagane parametry:

- Island – nazwa wyspy, z której pochodzi pingwin, dostępna w spisie wysp
- Bill_length_mm – długość dziobu podana w milimetrach
- Bill_depth_mm – głębokość dziobu podana w milimetrach
- Flipper_length_mm – długość płetwy podana w milimetrach

- Body_mass_g – masa ciała podana w gramach
- Sex – płeć

Opcjonalne parametry:

- Brak

Przykład wywołania:

/run-pipeline/save_data?island=Torgersen&bill_length_mm=39.1&bill_length_mm=18.7&flipper_length_mm=181.0&body_mass_g=3250.0&sex=0

{

„result”: „Adelie”

}