```
!pip install -r "C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt"
```

```
Requirement already satisfied: numpy==1.26.4 in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 1)) (1.26.4)
Requirement already satisfied: pandas in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 2)) (2.3.0)
Requirement already satisfied: scipy<1.14.0 in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 3)) (1.13.1)
Requirement already satisfied: xgboost in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 4)) (3.0.2)
Requirement already satisfied: lightgbm in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 5)) (4.6.0)
Requirement already satisfied: catboost in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 6)) (1.2.8)
Requirement already satisfied: tensorflow in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 7)) (2.19.0)
Requirement already satisfied: keras in c:\anaconda3\lib\site-packages
(from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 8)) (3.10.0)
Requirement already satisfied: torch in c:\anaconda3\lib\site-packages
(from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 9)) (2.7.1+cu118)
Requirement already satisfied: torchvision in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 10)) (0.22.1+cu118)
Requirement already satisfied: torchaudio in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 11)) (2.7.1+cu118)
Requirement already satisfied: spacy in c:\anaconda3\lib\site-packages
(from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 12)) (3.8.7)
Requirement already satisfied: nltk in c:\anaconda3\lib\site-packages
(from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 13)) (3.9.1)
Requirement already satisfied: beautifulsoup4 in c:\anaconda3\lib\
site-packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 14)) (4.12.3)
Requirement already satisfied: requests in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 15)) (2.32.4)
Requirement already satisfied: selenium in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
```

```
requirements.txt (line 16)) (4.34.2)
Requirement already satisfied: undetected-chromedriver in c:\
anaconda3\lib\site-packages (from -r C:\Users\aniru\OneDrive\Desktop\
ML tutorial\AIVA\requirements.txt (line 17)) (3.5.5)
Requirement already satisfied: mesa==1.2.1 in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 18)) (1.2.1)
Requirement already satisfied: deap in c:\anaconda3\lib\site-packages
(from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 19)) (1.4.3)
Requirement already satisfied: matplotlib in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 20)) (3.10.0)
Requirement already satisfied: seaborn in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 21)) (0.13.2)
Requirement already satisfied: plotly in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 22)) (6.0.1)
Requirement already satisfied: streamlit in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 23)) (1.46.0)
Requirement already satisfied: gradio in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 24)) (5.34.0)
Requirement already satisfied: tqdm in c:\anaconda3\lib\site-packages
(from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 25)) (4.67.1)
Requirement already satisfied: loguru in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 26)) (0.7.3)
Requirement already satisfied: pyyaml in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 27)) (6.0.2)
Requirement already satisfied: networkx in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 28)) (3.4.2)
Requirement already satisfied: pyro-ppl in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 29)) (1.9.1)
Requirement already satisfied: pymc in c:\anaconda3\lib\site-packages
(from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 30)) (5.25.1)
Requirement already satisfied: sentence-transformers in c:\anaconda3\
lib\site-packages (from -r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 31)) (4.1.0)
Requirement already satisfied: openai in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 32)) (1.97.1)
```

```
Requirement already satisfied: transformers in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 33)) (4.48.2)
Requirement already satisfied: accelerate in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 34)) (1.8.1)
Requirement already satisfied: datasets in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 35)) (4.0.0)
Requirement already satisfied: fsspec in c:\anaconda3\lib\site-
packages (from -r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 36)) (2025.3.0)
Requirement already satisfied: click in c:\anaconda3\lib\site-packages
(from mesa==1.2.1->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 18)) (8.1.8)
Requirement already satisfied: cookiecutter in c:\anaconda3\lib\site-
packages (from mesa==1.2.1->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 18)) (2.6.0)
Requirement already satisfied: tornado in c:\anaconda3\lib\site-
packages (from mesa==1.2.1->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 18)) (6.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\anaconda3\
lib\site-packages (from pandas->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 2)) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\anaconda3\lib\site-
packages (from pandas->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 2)) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\anaconda3\lib\
site-packages (from pandas->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 2)) (2025.2)
Requirement already satisfied: graphviz in c:\anaconda3\lib\site-
packages (from catboost->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 6)) (0.21)
Requirement already satisfied: six in c:\anaconda3\lib\site-packages
(from catboost->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 6)) (1.17.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\anaconda3\lib\
site-packages (from tensorflow->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 7)) (2.3.1)
Requirement already satisfied: astunparse>=1.6.0 in c:\anaconda3\lib\
site-packages (from tensorflow->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 7)) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in c:\anaconda3\
lib\site-packages (from tensorflow->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 7)) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
c:\anaconda3\lib\site-packages (from tensorflow->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 7)) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\anaconda3\
```

```
lib\site-packages (from tensorflow->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 7)) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in c:\anaconda3\lib\
site-packages (from tensorflow->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 7)) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\anaconda3\lib\
site-packages (from tensorflow->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 7)) (3.4.0)
Requirement already satisfied: packaging in c:\anaconda3\lib\site-
packages (from tensorflow->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 7)) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in c:\anaconda3\lib\site-
packages (from tensorflow->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 7)) (5.29.3)
Requirement already satisfied: setuptools in c:\anaconda3\lib\site-
packages (from tensorflow->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 7)) (78.1.1)
Requirement already satisfied: termcolor>=1.1.0 in c:\anaconda3\lib\
site-packages (from tensorflow->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 7)) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\
anaconda3\lib\site-packages (from tensorflow->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 7)) (4.14.1)
Requirement already satisfied: wrapt>=1.11.0 in c:\anaconda3\lib\site-
packages (from tensorflow->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 7)) (1.17.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\anaconda3\
lib\site-packages (from tensorflow->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 7)) (1.73.1)
Requirement already satisfied: tensorboard~=2.19.0 in c:\anaconda3\
lib\site-packages (from tensorflow->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 7)) (2.19.0)
Requirement already satisfied: h5py>=3.11.0 in c:\anaconda3\lib\site-
packages (from tensorflow->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 7)) (3.12.1)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in c:\
anaconda3\lib\site-packages (from tensorflow->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 7)) (0.5.1)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\
anaconda3\lib\site-packages (from requests->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 15)) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\anaconda3\lib\site-
packages (from requests->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 15)) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\anaconda3\lib\
site-packages (from requests->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 15)) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\anaconda3\lib\
```

```
site-packages (from requests->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 15)) (2025.7.14)
Requirement already satisfied: markdown>=2.6.8 in c:\anaconda3\lib\
site-packages (from tensorboard~=2.19.0->tensorflow->-r C:\Users\
aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 7))
(3.8)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in c:\anaconda3\lib\site-packages (from tensorboard~=2.19.0-
>tensorflow->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 7)) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\anaconda3\lib\
site-packages (from tensorboard~=2.19.0->tensorflow->-r C:\Users\
aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 7))
(3.1.3)
Requirement already satisfied: rich in c:\anaconda3\lib\site-packages
(from keras->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 8)) (13.9.4)
Requirement already satisfied: namex in c:\anaconda3\lib\site-packages
(from keras->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 8)) (0.1.0)
Requirement already satisfied: optree in c:\anaconda3\lib\site-
packages (from keras->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 8)) (0.16.0)
Requirement already satisfied: filelock in c:\anaconda3\lib\site-
packages (from torch->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 9)) (3.17.0)
Requirement already satisfied: sympy>=1.13.3 in c:\anaconda3\lib\site-
packages (from torch->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 9)) (1.13.3)
Requirement already satisfied: jinja2 in c:\anaconda3\lib\site-
packages (from torch->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 9)) (3.1.6)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in c:\anaconda3\
lib\site-packages (from torchvision->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 10)) (11.1.0)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in c:\
anaconda3\lib\site-packages (from spacy->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 12)) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in c:\
anaconda3\lib\site-packages (from spacy->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 12)) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in c:\
anaconda3\lib\site-packages (from spacy->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 12)) (1.0.13)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in c:\anaconda3\
lib\site-packages (from spacy->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 12)) (2.0.11)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in c:\anaconda3\
lib\site-packages (from spacy->-r C:\Users\aniru\OneDrive\Desktop\ML
```

```
tutorial\AIVA\requirements.txt (line 12)) (3.0.10)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in c:\anaconda3\
lib\site-packages (from spacy->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 12)) (8.3.4)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in c:\anaconda3\
lib\site-packages (from spacy->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 12)) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in c:\anaconda3\
lib\site-packages (from spacy->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 12)) (2.5.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in c:\
anaconda3\lib\site-packages (from spacy->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 12)) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in c:\anaconda3\
lib\site-packages (from spacy->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 12)) (0.4.1)
Requirement already satisfied: typer<1.0.0,>=0.3.0 in c:\anaconda3\
lib\site-packages (from spacy->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 12)) (0.16.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in
c:\anaconda3\lib\site-packages (from spacy->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 12)) (2.10.3)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in c:\
anaconda3\lib\site-packages (from spacy->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 12)) (3.5.0)
Requirement already satisfied: colorama in c:\anaconda3\lib\site-
packages (from tqdm->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 25)) (0.4.6)
Requirement already satisfied: language-data>=1.2 in c:\anaconda3\lib\
site-packages (from langcodes<4.0.0,>=3.2.0->spacy->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 12)) (1.3.0)
Requirement already satisfied: annotated-types>=0.6.0 in c:\anaconda3\
lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy->-
r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt
(line 12)) (0.6.0)
Requirement already satisfied: pydantic-core==2.27.1 in c:\anaconda3\
lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy->-
r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt
(line 12)) (2.27.1)
Requirement already satisfied: blis<1.3.0,>=1.2.0 in c:\anaconda3\lib\
site-packages (from thinc<8.4.0,>=8.3.4->spacy->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 12)) (1.2.1)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in c:\
anaconda3\lib\site-packages (from thinc<8.4.0,>=8.3.4->spacy->-r C:\
Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line
12)) (0.1.5)
Requirement already satisfied: shellingham>=1.3.0 in c:\anaconda3\lib\
site-packages (from typer<1.0.0,>=0.3.0->spacy->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 12)) (1.5.0)
```

```
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in c:\
anaconda3\lib\site-packages (from weasel<0.5.0,>=0.1.0->spacy->-r C:\
Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line
12)) (0.21.1)
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in c:\
anaconda3\lib\site-packages (from weasel<0.5.0,>=0.1.0->spacy->-r C:\
Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line
12)) (5.2.1)
Requirement already satisfied: joblib in c:\anaconda3\lib\site-
packages (from nltk->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 13)) (1.5.1)
Requirement already satisfied: regex>=2021.8.3 in c:\anaconda3\lib\
site-packages (from nltk->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 13)) (2024.11.6)
Requirement already satisfied: soupsieve>1.2 in c:\anaconda3\lib\site-
packages (from beautifulsoup4->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 14)) (2.5)
Requirement already satisfied: trio~=0.30.0 in c:\anaconda3\lib\site-
packages (from selenium->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 16)) (0.30.0)
Requirement already satisfied: trio-websocket~=0.12.2 in c:\anaconda3\
lib\site-packages (from selenium->-r C:\Users\aniru\OneDrive\Desktop\
ML tutorial\AIVA\requirements.txt (line 16)) (0.12.2)
Requirement already satisfied: websocket-client~=1.8.0 in c:\
anaconda3\lib\site-packages (from selenium->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 16)) (1.8.0)
Requirement already satisfied: attrs>=23.2.0 in c:\anaconda3\lib\site-
packages (from trio~=0.30.0->selenium->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 16)) (24.3.0)
Requirement already satisfied: sortedcontainers in c:\anaconda3\lib\
site-packages (from trio~=0.30.0->selenium->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 16)) (2.4.0)
Requirement already satisfied: outcome in c:\anaconda3\lib\site-
packages (from trio~=0.30.0->selenium->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 16)) (1.3.0.post0)
Requirement already satisfied: sniffio>=1.3.0 in c:\anaconda3\lib\
site-packages (from trio~=0.30.0->selenium->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 16)) (1.3.0)
Requirement already satisfied: cffi>=1.14 in c:\anaconda3\lib\site-
packages (from trio~=0.30.0->selenium->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 16)) (1.17.1)
Requirement already satisfied: wsproto>=0.14 in c:\anaconda3\lib\site-
packages (from trio-websocket~=0.12.2->selenium->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 16)) (1.2.0)
Requirement already satisfied: pysocks!=1.5.7,<2.0,>=1.5.6 in c:\
anaconda3\lib\site-packages (from urllib3[socks]~=2.5.0->selenium->-r
C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt
(line 16)) (1.7.1)
Requirement already satisfied: websockets in c:\anaconda3\lib\site-
```

```
packages (from undetected-chromedriver->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 17)) (15.0.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\anaconda3\lib\
site-packages (from matplotlib->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 20)) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\anaconda3\lib\site-
packages (from matplotlib->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 20)) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\anaconda3\lib\
site-packages (from matplotlib->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 20)) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\anaconda3\lib\
site-packages (from matplotlib->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 20)) (1.4.8)
Requirement already satisfied: pyparsing>=2.3.1 in c:\anaconda3\lib\
site-packages (from matplotlib->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 20)) (3.2.0)
Requirement already satisfied: narwhals>=1.15.1 in c:\anaconda3\lib\
site-packages (from plotly->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 22)) (1.31.0)
Requirement already satisfied: altair<6,>=4.0 in c:\anaconda3\lib\
site-packages (from streamlit->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 23)) (5.5.0)
Requirement already satisfied: blinker<2,>=1.5.0 in c:\anaconda3\lib\
site-packages (from streamlit->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 23)) (1.9.0)
Requirement already satisfied: cachetools<7,>=4.0 in c:\anaconda3\lib\
site-packages (from streamlit->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 23)) (5.5.1)
Requirement already satisfied: pyarrow>=7.0 in c:\anaconda3\lib\site-
packages (from streamlit->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 23)) (19.0.0)
Requirement already satisfied: tenacity<10,>=8.1.0 in c:\anaconda3\
lib\site-packages (from streamlit->-r C:\Users\aniru\OneDrive\Desktop\
ML tutorial\AIVA\requirements.txt (line 23)) (9.0.0)
Requirement already satisfied: toml<2,>=0.10.1 in c:\anaconda3\lib\
site-packages (from streamlit->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 23)) (0.10.2)
Requirement already satisfied: watchdog<7,>=2.1.5 in c:\anaconda3\lib\
site-packages (from streamlit->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 23)) (4.0.2)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in c:\
anaconda3\lib\site-packages (from streamlit->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 23)) (3.1.43)
Requirement already satisfied: jsonschema>=3.0 in c:\anaconda3\lib\
site-packages (from altair<6,>=4.0->streamlit->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 23)) (4.23.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in c:\anaconda3\lib\
site-packages (from gitpython!=3.1.19,<4,>=3.0.7->streamlit->-r C:\
```

```
Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line
23)) (4.0.7)
Requirement already satisfied: smmap<5,>=3.0.1 in c:\anaconda3\lib\
site-packages (from gitdb<5,>=4.0.1->gitpython!=3.1.19,<4,>=3.0.7-
>streamlit->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 23)) (4.0.0)
Requirement already satisfied: aiofiles<25.0,>=22.0 in c:\anaconda3\
lib\site-packages (from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 24)) (24.1.0)
Requirement already satisfied: anyio<5.0,>=3.0 in c:\anaconda3\lib\
site-packages (from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 24)) (4.7.0)
Requirement already satisfied: fastapi<1.0,>=0.115.2 in c:\anaconda3\
lib\site-packages (from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 24)) (0.115.13)
Requirement already satisfied: ffmpy in c:\anaconda3\lib\site-packages
(from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 24)) (0.6.0)
Requirement already satisfied: gradio-client==1.10.3 in c:\anaconda3\
lib\site-packages (from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 24)) (1.10.3)
Requirement already satisfied: groovy~=0.1 in c:\anaconda3\lib\site-
packages (from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 24)) (0.1.2)
Requirement already satisfied: httpx>=0.24.1 in c:\anaconda3\lib\site-
packages (from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 24)) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.28.1 in c:\
anaconda3\lib\site-packages (from gradio->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 24)) (0.32.3)
Requirement already satisfied: markupsafe<4.0,>=2.0 in c:\anaconda3\
lib\site-packages (from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 24)) (3.0.2)
Requirement already satisfied: orjson~=3.0 in c:\anaconda3\lib\site-
packages (from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 24)) (3.10.18)
Requirement already satisfied: pydub in c:\anaconda3\lib\site-packages
(from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 24)) (0.25.1)
Requirement already satisfied: python-multipart>=0.0.18 in c:\
anaconda3\lib\site-packages (from gradio->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 24)) (0.0.20)
Requirement already satisfied: ruff>=0.9.3 in c:\anaconda3\lib\site-
packages (from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 24)) (0.12.0)
Requirement already satisfied: safehttpx<0.2.0,>=0.1.6 in c:\
anaconda3\lib\site-packages (from gradio->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 24)) (0.1.6)
Requirement already satisfied: semantic-version~=2.0 in c:\anaconda3\
```

```
lib\site-packages (from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 24)) (2.10.0)
Requirement already satisfied: starlette<1.0,>=0.40.0 in c:\anaconda3\
lib\site-packages (from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 24)) (0.46.2)
Requirement already satisfied: tomlkit<0.14.0,>=0.12.0 in c:\
anaconda3\lib\site-packages (from gradio->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 24)) (0.13.2)
Requirement already satisfied: uvicorn>=0.14.0 in c:\anaconda3\lib\
site-packages (from gradio->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 24)) (0.34.3)
Requirement already satisfied: win32-setctime>=1.0.0 in c:\anaconda3\
lib\site-packages (from loguru->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 26)) (1.2.0)
Requirement already satisfied: pyro-api>=0.1.1 in c:\anaconda3\lib\
site-packages (from pyro-ppl->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 29)) (0.1.2)
Requirement already satisfied: arviz>=0.13.0 in c:\anaconda3\lib\site-
packages (from pymc->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 30)) (0.22.0)
Requirement already satisfied: cloudpickle in c:\anaconda3\lib\site-
packages (from pymc->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\
AIVA\requirements.txt (line 30)) (3.0.0)
Requirement already satisfied: pytensor<2.32,>=2.31.7 in c:\anaconda3\
lib\site-packages (from pymc->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 30)) (2.31.7)
Requirement already satisfied: threadpoolctl<4.0.0,>=3.1.0 in c:\
anaconda3\lib\site-packages (from pymc->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 30)) (3.5.0)
Requirement already satisfied: etuples in c:\anaconda3\lib\site-
packages (from pytensor<2.32,>=2.31.7->pymc->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 30)) (0.3.10)
Requirement already satisfied: logical-unification in c:\anaconda3\
lib\site-packages (from pytensor<2.32,>=2.31.7->pymc->-r C:\Users\
aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 30))
(0.4.6)
Requirement already satisfied: miniKanren in c:\anaconda3\lib\site-
packages (from pytensor<2.32,>=2.31.7->pymc->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 30)) (1.0.5)
Requirement already satisfied: cons in c:\anaconda3\lib\site-packages
(from pytensor<2.32,>=2.31.7->pymc->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 30)) (0.4.7)
Requirement already satisfied: scikit-learn in c:\anaconda3\lib\site-
packages (from sentence-transformers->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 31)) (1.6.1)
Requirement already satisfied: tokenizers<0.22,>=0.21 in c:\anaconda3\
lib\site-packages (from transformers->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 33)) (0.21.1)
Requirement already satisfied: safetensors>=0.4.1 in c:\anaconda3\lib\
```

```
site-packages (from transformers->-r C:\Users\aniru\OneDrive\Desktop\
ML tutorial\AIVA\requirements.txt (line 33)) (0.5.3)
Requirement already satisfied: distro<2,>=1.7.0 in c:\anaconda3\lib\
site-packages (from openai->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 32)) (1.9.0)
Requirement already satisfied: jiter<1,>=0.4.0 in c:\anaconda3\lib\
site-packages (from openai->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 32)) (0.10.0)
Requirement already satisfied: httpcore==1.* in c:\anaconda3\lib\site-
packages (from httpx>=0.24.1->gradio->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 24)) (1.0.9)
Requirement already satisfied: h11>=0.16 in c:\anaconda3\lib\site-
packages (from httpcore==1.*->httpx>=0.24.1->gradio->-r C:\Users\
aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 24))
(0.16.0)
Requirement already satisfied: psutil in c:\anaconda3\lib\site-
packages (from accelerate->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 34)) (5.9.0)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in c:\anaconda3\lib\
site-packages (from datasets->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 35)) (0.3.8)
Requirement already satisfied: xxhash in c:\anaconda3\lib\site-
packages (from datasets->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 35)) (3.5.0)
Requirement already satisfied: multiprocess<0.70.17 in c:\anaconda3\
lib\site-packages (from datasets->-r C:\Users\aniru\OneDrive\Desktop\
ML tutorial\AIVA\requirements.txt (line 35)) (0.70.16)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in c:\
anaconda3\lib\site-packages (from fsspec[http]<=2025.3.0,>=2023.1.0-
>datasets->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 35)) (3.11.10)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in c:\
anaconda3\lib\site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1-
>fsspec[http]<=2025.3.0,>=2023.1.0->datasets->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 35)) (2.4.4)
Requirement already satisfied: aiosignal>=1.1.2 in c:\anaconda3\lib\
site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1-
>fsspec[http]<=2025.3.0,>=2023.1.0->datasets->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 35)) (1.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in c:\anaconda3\lib\
site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1-
>fsspec[http]<=2025.3.0,>=2023.1.0->datasets->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 35)) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in c:\anaconda3\
lib\site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1-
>fsspec[http]<=2025.3.0,>=2023.1.0->datasets->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 35)) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in c:\anaconda3\lib\
site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1-
>fsspec[http]<=2025.3.0,>=2023.1.0->datasets->-r C:\Users\aniru\
```

```
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 35)) (0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in c:\anaconda3\lib\
site-packages (from aiohttp!=4.0.0a0,!=4.0.0a1-
>fsspec[http]<=2025.3.0,>=2023.1.0->datasets->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 35)) (1.18.0)
Requirement already satisfied: xarray>=2023.7.0 in c:\anaconda3\lib\
site-packages (from arviz>=0.13.0->pymc->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 30)) (2025.4.0)
Requirement already satisfied: h5netcdf>=1.0.2 in c:\anaconda3\lib\
site-packages (from arviz>=0.13.0->pymc->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 30)) (1.6.3)
Requirement already satisfied: xarray-einstats>=0.3 in c:\anaconda3\
lib\site-packages (from arviz>=0.13.0->pymc->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 30)) (0.9.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\anaconda3\lib\
site-packages (from astunparse>=1.6.0->tensorflow->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 7)) (0.45.1)
Requirement already satisfied: pycparser in c:\anaconda3\lib\site-
packages (from cffi>=1.14->trio~=0.30.0->selenium->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 16)) (2.21)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
c:\anaconda3\lib\site-packages (from jsonschema>=3.0->altair<6,>=4.0-
>streamlit->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 23)) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in c:\anaconda3\
lib\site-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit->-r
C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt
(line 23)) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in c:\anaconda3\lib\
site-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit->-r C:\
Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line
23)) (0.22.3)
Requirement already satisfied: marisa-trie>=1.1.0 in c:\anaconda3\lib\
site-packages (from language-data>=1.2->langcodes<4.0.0,>=3.2.0-
>spacy->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 12)) (1.2.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\anaconda3\
lib\site-packages (from rich->keras->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 8)) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\
anaconda3\lib\site-packages (from rich->keras->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 8)) (2.19.1)
Requirement already satisfied: mdurl~=0.1 in c:\anaconda3\lib\site-
packages (from markdown-it-py>=2.2.0->rich->keras->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 8)) (0.1.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\anaconda3\lib\
site-packages (from sympy>=1.13.3->torch->-r C:\Users\aniru\OneDrive\
Desktop\ML tutorial\AIVA\requirements.txt (line 9)) (1.3.0)
Requirement already satisfied: toolz in c:\anaconda3\lib\site-packages
```

```
(from logical-unification->pytensor<2.32,>=2.31.7->pymc->-r C:\Users\
aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 30))
(1.0.0)
Requirement already satisfied: multipledispatch in c:\anaconda3\lib\
site-packages (from logical-unification->pytensor<2.32,>=2.31.7->pymc-
>-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt
(line 30)) (0.6.0)
Requirement already satisfied: binaryornot>=0.4.4 in c:\anaconda3\lib\
site-packages (from cookiecutter->mesa==1.2.1->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 18)) (0.4.4)
Requirement already satisfied: python-slugify>=4.0.0 in c:\anaconda3\
lib\site-packages (from cookiecutter->mesa==1.2.1->-r C:\Users\aniru\
OneDrive\Desktop\ML tutorial\AIVA\requirements.txt (line 18)) (5.0.2)
Requirement already satisfied: arrow in c:\anaconda3\lib\site-packages
(from cookiecutter->mesa==1.2.1->-r C:\Users\aniru\OneDrive\Desktop\ML
tutorial\AIVA\requirements.txt (line 18)) (1.3.0)
Requirement already satisfied: chardet>=3.0.2 in c:\anaconda3\lib\
site-packages (from binaryornot>=0.4.4->cookiecutter->mesa==1.2.1->-r
C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\requirements.txt
(line 18)) (4.0.0)
Requirement already satisfied: text-unidecode>=1.3 in c:\anaconda3\
lib\site-packages (from python-slugify>=4.0.0->cookiecutter-
>mesa==1.2.1->-r C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
requirements.txt (line 18)) (1.3)

!pip install pandas numpy scikit-learn tqdm faker beautifulsoup4
requests pyarrow

Requirement already satisfied: pandas in c:\anaconda3\lib\site-
packages (2.3.0)
Requirement already satisfied: numpy in c:\anaconda3\lib\site-packages
(1.26.4)
Requirement already satisfied: scikit-learn in c:\anaconda3\lib\site-
packages (1.6.1)
Requirement already satisfied: tqdm in c:\anaconda3\lib\site-packages
(4.67.1)
Requirement already satisfied: faker in c:\anaconda3\lib\site-packages
(37.5.3)
Requirement already satisfied: beautifulsoup4 in c:\anaconda3\lib\
site-packages (4.12.3)
Requirement already satisfied: requests in c:\anaconda3\lib\site-
packages (2.32.4)
Requirement already satisfied: pyarrow in c:\anaconda3\lib\site-
packages (19.0.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\anaconda3\
lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\anaconda3\lib\site-
packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\anaconda3\lib\
site-packages (from pandas) (2025.2)
```

```
Requirement already satisfied: scipy>=1.6.0 in c:\anaconda3\lib\site-
packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in c:\anaconda3\lib\site-
packages (from scikit-learn) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\anaconda3\
lib\site-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: colorama in c:\anaconda3\lib\site-
packages (from tqdm) (0.4.6)
Requirement already satisfied: soupsieve>1.2 in c:\anaconda3\lib\site-
packages (from beautifulsoup4) (2.5)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\
anaconda3\lib\site-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\anaconda3\lib\site-
packages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\anaconda3\lib\
site-packages (from requests) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\anaconda3\lib\
site-packages (from requests) (2025.7.14)
Requirement already satisfied: six>=1.5 in c:\anaconda3\lib\site-
packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

```
import os, math, random, warnings, sys, time, pickle, urllib.parse,
datetime as dt
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, roc_auc_score
from sklearn.utils import resample
from tqdm import tqdm
from bs4 import BeautifulSoup
import requests

USE_SELENIUM = False
if USE_SELENIUM:
    try:
        from selenium import webdriver
        from selenium.webdriver.chrome.options import Options
    except Exception:
        print("[WARN] Selenium not available. Set USE_SELENIUM=False
or install it.")

DATA_ROOT = r"C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
datasets"
OUT_ROOT  = r"C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA"
os.makedirs(OUT_ROOT, exist_ok=True)
```

```
WRITE_PARQUET = False
USE_OHE = False
SAVE_ENCODERS = True
SYNTHETIC_N  = 50000
RF_WARM_BATCH = 5000
PRED_THRESH   = 0.30
RAND_SEED     = 42
N_AGENTS = 20
N_GENERATIONS = 10
TOP_K = 6
N_CHILDREN = N_AGENTS - TOP_K
AGENT_BATCH = 5000
MUTATION_RATE = 0.2
GA_REAL_RATIO = 0.7
N_META = 20000
np.random.seed(RAND_SEED)
random.seed(RAND_SEED)

def make_org_key(df, col):
    return df[col].astype(str).str.lower().str.strip()
def safe_read_csv(path, **kw):
    if not os.path.exists(path):
        print(f"[WARN] Missing: {path}"); return pd.DataFrame()
    try:
        return pd.read_csv(path, **kw)
    except Exception as e:
        print(f"[WARN] CSV read fail {path}: {e}"); return
pd.DataFrame()
def safe_read_excel(path, **kw):
    if not os.path.exists(path):
        print(f"[WARN] Missing: {path}"); return pd.DataFrame()
    try:
        return pd.read_excel(path, **kw)
    except Exception as e:
        print(f"[WARN] Excel read fail {path}: {e}"); return
pd.DataFrame()
def normalize_list_like_series(s: pd.Series) -> pd.Series:
    return (
        s.astype(str)
         .str.replace(r'[\[\]\']', '', regex=True)
         .str.replace(r'\s+', ' ', regex=True)
         .str.strip()
         .replace({'nan': np.nan, 'None': np.nan, '': np.nan})
    )
def to_io(df, path_stem):
    if WRITE_PARQUET:
        p = os.path.join(OUT_ROOT, path_stem + ".parquet")
        df.to_parquet(p, index=False)
    else:
        p = os.path.join(OUT_ROOT, path_stem + ".csv")
```

```
        df.to_csv(p, index=False)
    return p
def from_io(path_stem):
    p_parq = os.path.join(OUT_ROOT, path_stem + ".parquet")
    p_csv  = os.path.join(OUT_ROOT, path_stem + ".csv")
    if os.path.exists(p_parq):
        return pd.read_parquet(p_parq)
    return pd.read_csv(p_csv, low_memory=False)

ROUND_TO_STAGE = {
    'angel': 'Seed',
    'seed': 'Seed',
    'pre-seed': 'Seed',
    'series_a': 'Early',
    'a': 'Early',
    'series_b': 'Growth',
    'b': 'Growth',
    'series_c': 'Growth',
    'c': 'Growth',
    'mezzanine': 'Late',
    'late': 'Late',
    'private_equity': 'Late',
    'ipo': 'Exit',
    'acquired': 'Exit'
}
def map_company_stage(latest_round_type: pd.Series) -> pd.Series:
    def _m(x):
        if pd.isna(x): return "unknown"
        s = str(x).strip().lower().replace(" ", "_")
        return ROUND_TO_STAGE.get(s, "unknown")
    return latest_round_type.apply(_m)

def fetch_news_features_for_company(name, days_recent=30,
max_items=200):
    """Return (news_count_all, latest_news_date, news_count_recent)
using Google News RSS."""
    if not isinstance(name, str) or not name.strip():
        return 0, pd.NaT, 0
    q = urllib.parse.quote_plus(name.strip())
    url = f"https://news.google.com/rss/search?q={q}&hl=en-
US&gl=US&ceid=US:en"
    try:
        r = requests.get(url, timeout=10)
        if r.status_code != 200:
            return 0, pd.NaT, 0
        soup = BeautifulSoup(r.content, "xml")
        items = soup.find_all("item")
        total = min(len(items), max_items)
        latest_date = pd.NaT
        recent_count = 0
```

```
        cutoff = pd.Timestamp.utcnow() -
pd.Timedelta(days=days_recent)
        for it in items[:max_items]:
            pub = it.find("pubDate")
            if pub and pub.text:
                try:
                    dt_parsed = pd.to_datetime(pub.text,
errors='coerce', utc=True)
                except Exception:
                    dt_parsed = pd.NaT
                if pd.notna(dt_parsed):
                    if pd.isna(latest_date) or dt_parsed >
latest_date:
                        latest_date = dt_parsed
                    if dt_parsed >= cutoff:
                        recent_count += 1
        return total, latest_date, recent_count
    except Exception:
        return 0, pd.NaT, 0

def selenium_fetch_title(url):
    if not USE_SELENIUM:
        return None
    try:
        options = Options()
        options.add_argument("--headless=new")
        options.add_argument("--no-sandbox")
        options.add_argument("--disable-gpu")
        driver = webdriver.Chrome(options=options)
        driver.get(url)
        time.sleep(2)
        title = driver.title
        driver.quit()
        return title
    except Exception:
        return None

acq   = safe_read_csv(os.path.join(DATA_ROOT, "acquisitions.csv"))
deg   = safe_read_csv(os.path.join(DATA_ROOT, "degrees.csv"))
funds = safe_read_csv(os.path.join(DATA_ROOT, "funds.csv"))
fr    = safe_read_csv(os.path.join(DATA_ROOT, "funding_rounds.csv"))
inv   = safe_read_csv(os.path.join(DATA_ROOT, "investments.csv"))
ipos  = safe_read_csv(os.path.join(DATA_ROOT, "ipos.csv"))
mile  = safe_read_csv(os.path.join(DATA_ROOT, "milestones.csv"))
obj   = safe_read_csv(os.path.join(DATA_ROOT, "objects.csv"),
dtype=str)
off   = safe_read_csv(os.path.join(DATA_ROOT, "offices.csv"))
people= safe_read_csv(os.path.join(DATA_ROOT, "people.csv"))
rel   = safe_read_csv(os.path.join(DATA_ROOT, "relationships.csv"))
yc    = safe_read_csv(os.path.join(DATA_ROOT, "yc_companies.csv"))
```

```
invvc = safe_read_csv(os.path.join(DATA_ROOT, "investments_VC.csv"),
encoding='latin1')
yc_clean = safe_read_excel(os.path.join(DATA_ROOT,
"yc_cleaned_data.xlsx"))

if not obj.empty and 'id' in obj.columns: obj['org_key'] =
make_org_key(obj, 'id')
if not fr.empty and 'object_id' in fr.columns: fr['org_key'] =
make_org_key(fr, 'object_id')
if not acq.empty and 'acquired_object_id' in acq.columns:
acq['org_key'] = make_org_key(acq, 'acquired_object_id')
if not ipos.empty and 'object_id' in ipos.columns: ipos['org_key'] =
make_org_key(ipos, 'object_id')
if not off.empty and 'object_id' in off.columns: off['org_key'] =
make_org_key(off, 'object_id')
if not mile.empty and 'object_id' in mile.columns: mile['org_key'] =
make_org_key(mile, 'object_id')
if not people.empty and 'object_id' in people.columns:
people['org_key'] = make_org_key(people, 'object_id')
if not rel.empty and 'relationship_object_id' in rel.columns:
rel['org_key'] = make_org_key(rel, 'relationship_object_id')
if not inv.empty and 'funded_object_id' in inv.columns: inv['org_key']
= make_org_key(inv, 'funded_object_id')
if not yc_clean.empty:
    if 'permalink' in yc_clean.columns:
        yc_clean['org_key'] = make_org_key(yc_clean, 'permalink')
    elif 'name' in yc_clean.columns:
        yc_clean['org_key'] =
yc_clean['name'].astype(str).str.lower().str.strip()

if not off.empty:
    if 'description' in off.columns:
        hq =
(off[off['description'].astype(str).str.lower().str.contains('head',
na=False)]
                .groupby('org_key').first().reset_index())
        if hq.empty: hq = off.groupby('org_key').first().reset_index()
    else:
        hq = off.groupby('org_key').first().reset_index()
    hq = hq.rename(columns={'city': 'hq_city', 'state_code':
'hq_state', 'country_code': 'hq_country'})
else:
    hq = pd.DataFrame(columns=['org_key', 'hq_city', 'hq_state',
'hq_country'])

if not fr.empty:
    fr['funded_at'] = pd.to_datetime(fr.get('funded_at', pd.NaT),
errors='coerce')
    fr = fr.sort_values(['org_key', 'funded_at'])
    funding = fr.groupby('org_key').agg(
```

```
        total_raised_usd=('raised_amount_usd', 'sum'),
        num_rounds=('funding_round_code', 'count'),
        latest_round_type=('funding_round_code', 'last'),
        first_funding_at=('funded_at', 'min'),
        last_funding_at=('funded_at', 'max'),
        avg_round_size_usd=('raised_amount_usd', 'mean'),
        funding_round_codes=('funding_round_code', lambda x:
','.join(sorted(set(x.dropna().astype(str)))))
    ).reset_index()
    dur_years = ((funding['last_funding_at'] -
funding['first_funding_at']).dt.days / 365.25).replace(0, np.nan)
    funding['funding_per_year'] = funding['total_raised_usd'] /
dur_years
    funding['company_stage'] =
map_company_stage(funding['latest_round_type'])
else:
    funding = pd.DataFrame(columns=[
        'org_key', 'total_raised_usd', 'num_rounds',
'latest_round_type',
        'first_funding_at', 'last_funding_at', 'avg_round_size_usd',
        'funding_round_codes', 'funding_per_year', 'company_stage'
    ])

if not inv.empty and 'investor_object_id' in inv.columns:
    investors_count = inv.groupby('org_key')
['investor_object_id'].nunique().reset_index().rename(columns={'invest
or_object_id': 'investors_count'})
else:
    investors_count = pd.DataFrame(columns=['org_key',
'investors_count'])
if not inv.empty:
    if 'is_lead_investor' in inv.columns:
        lead_investors = (inv[inv['is_lead_investor'] == 1]
                          .groupby('org_key')['investor_object_id']
                          .nunique().rename('lead_investors').reset_in
dex())
    else:
        lead_investors = (inv.groupby('org_key')['investor_object_id']
                          .nunique().rename('lead_investors').reset_in
dex())
else:
    lead_investors = pd.DataFrame(columns=['org_key',
'lead_investors'])

exit_flag = pd.DataFrame(columns=['org_key', 'exit_flag'])
ipo_at = pd.DataFrame(columns=['org_key', 'ipo_at'])
acquired_at = pd.DataFrame(columns=['org_key', 'acquired_at'])

if not acq.empty:
    acquired_at = acq[['org_key',
```

```
'acquired_at']].dropna(subset=['org_key']).drop_duplicates()
if not ipos.empty:
    ipo_at = ipos[['org_key',
'public_at']].dropna(subset=['org_key']).drop_duplicates().rename(colu
mns={'public_at': 'ipo_at'})
if (not acq.empty) or (not ipos.empty):
    ef = pd.concat([
        acq[['org_key']] if 'org_key' in acq else
pd.DataFrame(columns=['org_key']),
        ipos[['org_key']] if 'org_key' in ipos else
pd.DataFrame(columns=['org_key'])
    ], ignore_index=True).drop_duplicates()
    ef['exit_flag'] = 1
    exit_flag = ef

if not mile.empty:
    milestones_count =
mile.groupby('org_key').size().rename('milestones_count').reset_index(
)
    milestone_latest = mile.groupby('org_key')
['milestone_at'].max().reset_index().rename(columns={'milestone_at':
'milestone_latest'})
else:
    milestones_count = pd.DataFrame(columns=['org_key',
'milestones_count'])
    milestone_latest = pd.DataFrame(columns=['org_key',
'milestone_latest'])

founders = pd.DataFrame(columns=['org_key', 'num_founders'])
founder_education = pd.DataFrame(columns=['org_key',
'founder_education'])
founder_universities = pd.DataFrame(columns=['org_key',
'founder_universities'])

if not rel.empty:
    founder_rels =
rel[rel['title'].astype(str).str.lower().str.contains('founder',
na=False)]
    if not founder_rels.empty:
        founders = (founder_rels.groupby('org_key')
['person_object_id']
                    .nunique().reset_index().rename(columns={'person_o
bject_id': 'num_founders'}))
        if not deg.empty:
            deg = deg.rename(columns={'object_id': 'person_key'})
            deg_founders = pd.merge(
                founder_rels[['org_key', 'person_object_id']],
                deg,
                left_on='person_object_id',
                right_on='person_key',
```

```
                how='left'
            )
            founder_education = (deg_founders.groupby('org_key')
['degree_type']
                                 .apply(lambda x:
','.join(sorted(set([str(v) for v in x.dropna()]))))
                                 .reset_index().rename(columns={'degre
e_type': 'founder_education'}))
            founder_universities = (deg_founders.groupby('org_key')
['institution']
                                    .apply(lambda x:
','.join(sorted(set([str(v) for v in x.dropna()]))))
                                    .reset_index().rename(columns={'in
stitution': 'founder_universities'}))

yc_subset = pd.DataFrame(columns=['org_key'])
if not yc_clean.empty and 'org_key' in yc_clean.columns:
    keep_cols = [c for c in ["yc_batch", "yc_batch_year", "yc_status",
"yc_top_company", "is_unicorn"] if c in yc_clean.columns]
    yc_subset = yc_clean[['org_key'] + keep_cols].copy()

vc_counts = pd.DataFrame(columns=['org_key', 'vc_deal_count',
'has_vc_backing'])
if not invvc.empty:
    invvc['org_key'] = invvc.get('permalink',
'').astype(str).str.lower().str.strip()
    tmp =
invvc.groupby('org_key').size().rename('vc_deal_count').reset_index()
    tmp['has_vc_backing'] = 1
    vc_counts = tmp

funds_feat = pd.DataFrame(columns=['org_key', 'num_funds',
'funds_total_usd', 'latest_fund_raised_at'])
if not funds.empty:
    if 'object_id' in funds.columns:
        funds['org_key'] = make_org_key(funds, 'object_id')
    elif 'permalink' in funds.columns:
        funds['org_key'] = make_org_key(funds, 'permalink')
    else:
        funds['org_key'] = np.nan
    amount_candidates = [
        'raised_amount_usd', 'raised_amount', 'fund_size_usd',
'fund_size',
        'capital_committed_usd', 'capital_committed'
    ]
    date_candidates = ['raised_at', 'announced_on', 'closed_on',
'created_at']
    amount_col = next((c for c in amount_candidates if c in
funds.columns), None)
    date_col   = next((c for c in date_candidates   if c in
```

```
funds.columns), None)
    if amount_col:
        funds[amount_col] = pd.to_numeric(funds[amount_col],
errors='coerce')
    if date_col:
        funds[date_col] = pd.to_datetime(funds[date_col],
errors='coerce', utc=True)
    agg_dict = {'num_funds': (amount_col if amount_col else (date_col
if date_col else 'org_key'), 'count')}
    if amount_col:
        agg_dict['funds_total_usd'] = (amount_col, 'sum')
    if date_col:
        agg_dict['latest_fund_raised_at'] = (date_col, 'max')
    if 'org_key' in funds.columns:
        funds_feat = (
            funds.dropna(subset=['org_key'])
                 .groupby('org_key')
                 .agg(**agg_dict)
                 .reset_index()
        )
    for col in
['num_funds','funds_total_usd','latest_fund_raised_at']:
        if col not in funds_feat.columns:
            funds_feat[col] = 0 if col != 'latest_fund_raised_at' else
pd.NaT

if obj.empty:
    raise ValueError("objects.csv missing/empty; cannot build
master.")
obj['founded_year'] = pd.to_datetime(obj.get('founded_at', pd.NaT),
errors='coerce').dt.year
obj['company_age']  = 2025 - obj['founded_year'].fillna(2025)
obj['funding_total_usd'] = pd.to_numeric(obj.get('funding_total_usd',
0), errors='coerce')
obj['log_funding']  = np.log10(obj['funding_total_usd'].fillna(0) + 1)
main_cols =
["name","org_key","category_code","status","founded_at","founded_year"
,"company_age","log_funding","funding_total_usd"]
master = obj[[c for c in main_cols if c in obj.columns]].copy()
num_offices =
(off.groupby('org_key').size().reset_index(name='num_offices')) if not
off.empty else pd.DataFrame(columns=['org_key','num_offices'])
num_investors = investors_count.copy()

MAX_NEWS_FETCH = 500
sampled_obj = obj[['org_key','name']].dropna().sample(MAX_NEWS_FETCH,
random_state=42)

import time, functools
```

```
def fetch_news_features_for_company_cached(name, days_recent=30,
max_items=200):
    total, latest_dt, recent = fetch_news_features_for_company(name,
days_recent, max_items)
    time.sleep(0.5)
    return total, latest_dt, recent
news_df = pd.DataFrame({
    'org_key': obj['org_key'],
    'news_count': 0,
    'latest_news_date': pd.NaT,
    'recent_news_count': 0
})

master = (master
    .merge(hq[['org_key','hq_city','hq_state','hq_country']],
on='org_key', how='left')
    .merge(num_offices, on='org_key', how='left')
    .merge(funding, on='org_key', how='left')
    .merge(lead_investors, on='org_key', how='left')
    .merge(investors_count, on='org_key', how='left')
    .merge(exit_flag[['org_key','exit_flag']], on='org_key',
how='left')
    .merge(ipo_at, on='org_key', how='left')
    .merge(acquired_at, on='org_key', how='left')
    .merge(milestones_count, on='org_key', how='left')
    .merge(milestone_latest, on='org_key', how='left')
    .merge(founders, on='org_key', how='left')
    .merge(pd.DataFrame({'org_key': master['org_key'],
'num_employees': np.nan}), on='org_key', how='left')
    .merge(pd.DataFrame({'org_key': master['org_key'],
'num_current_employees': np.nan}), on='org_key', how='left')
    .merge(founder_education, on='org_key', how='left')
    .merge(founder_universities, on='org_key', how='left')
    .merge(num_investors, on='org_key', how='left',
suffixes=('','_dup'))
    .merge(news_df, on='org_key', how='left')
    .merge(yc_subset, on='org_key', how='left')
    .merge(vc_counts, on='org_key', how='left')
    .merge(funds_feat, on='org_key', how='left')
)
if 'investors_count_dup' in master.columns:
    master.drop(columns=['investors_count_dup'], inplace=True,
errors='ignore')

master_cols = [

"name","org_key","category_code","category_list","company_stage","stat
us","founded_at","founded_year",
    "hq_city","hq_state","hq_country","num_offices","company_age",
```

```
"funding_total_usd","total_raised_usd","num_rounds","latest_round_type
","funding_per_year","log_funding",

"first_funding_at","last_funding_at","funding_round_codes","avg_round_
size_usd","lead_investors","investors_count",

"exit_flag","ipo_at","acquired_at","milestones_count","milestone_lates
t",

"num_founders","num_employees","num_current_employees","founder_educat
ion","founder_universities","num_investors",
    "news_count","latest_news_date","recent_news_count",

"yc_batch","yc_batch_year","yc_status","yc_top_company","has_vc_backin
g","vc_deal_count","is_unicorn",
    "num_funds","funds_total_usd","latest_fund_raised_at"
]
for c in master_cols:
    if c not in master.columns: master[c] = np.nan
master = master[master_cols].drop_duplicates('org_key')

if 'category_list' in master.columns:
    master['category_list'] =
normalize_list_like_series(master['category_list'])

for col in master.select_dtypes(include='object').columns:
    master[col] = (master[col].astype(str)
                   .str.strip()
                   .str.replace(r'\s+', ' ', regex=True)
                   .replace({'nan': np.nan, 'None': np.nan, '':
np.nan}))

num_cols = [

'company_age','num_offices','funding_total_usd','total_raised_usd','nu
m_rounds','funding_per_year','log_funding',

'avg_round_size_usd','investors_count','num_investors','num_founders',
'num_employees','num_current_employees',

'milestones_count','exit_flag','news_count','recent_news_count','has_v
c_backing','is_unicorn','yc_batch_year',
    'vc_deal_count','yc_top_company','num_funds','funds_total_usd'
]
for col in num_cols:
    master[col] = pd.to_numeric(master[col], errors='coerce')
date_cols =
['founded_at','first_funding_at','last_funding_at','ipo_at','acquired_
at','milestone_latest','latest_news_date','latest_fund_raised_at']
for col in date_cols:
```

```
    master[col] = pd.to_datetime(master[col], errors='coerce',
utc=True)
if 'company_age' in master.columns:
    master['company_age'] = master['company_age'].clip(lower=0)

cat_fill =
['category_code','category_list','company_stage','status','hq_city','h
q_state','hq_country',

'latest_round_type','lead_investors','founder_education','founder_univ
ersities','yc_batch','yc_status']
for col in cat_fill:
    master[col] = master[col].replace('nan', np.nan).fillna("unknown")

fill0_cols =
['company_age','num_offices','funding_total_usd','total_raised_usd','n
um_rounds','funding_per_year','log_funding',

'avg_round_size_usd','investors_count','num_investors','num_founders',
'num_employees','num_current_employees',

'milestones_count','exit_flag','news_count','recent_news_count','has_v
c_backing','is_unicorn','yc_batch_year',

'vc_deal_count','yc_top_company','num_funds','funds_total_usd']
for col in fill0_cols:
    master[col] = master[col].fillna(0)

if ('exit_flag' not in master.columns) or
master['exit_flag'].isnull().all():
    statuses = master.get('status',
pd.Series(dtype=str)).astype(str).str.lower()
    master['exit_flag'] = (
        statuses.isin(['exited','acquired','ipo'])
        | master.get('ipo_at',
pd.Series(index=master.index)).notnull()
        | master.get('acquired_at',
pd.Series(index=master.index)).notnull()
    ).astype(int)

p_master_raw = to_io(master, "master_startup_dataset")
thresh = int(0.8 * master.shape[1])
master = master.dropna(thresh=thresh).reset_index(drop=True)
p_master_clean = to_io(master, "master_startup_dataset_clean")
print(f"[OK] Wrote master datasets:\n- {p_master_raw}\n-
{p_master_clean}")
```

```
[OK] Wrote master datasets:
- C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
master_startup_dataset.csv
```

```
- C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
master_startup_dataset_clean.csv

print("\n[MASTER INFO]")
print(master.info())
print("\n[MASTER DESCRIBE]")
print(master.describe(include='all').T)
print("\n[MASTER NULLS TOP-20]")
print(master.isnull().sum().sort_values(ascending=False).head(20))


[MASTER INFO]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 462648 entries, 0 to 462647
Data columns (total 49 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   name                    462647 non-null  object
 1   org_key                 462648 non-null  object
 2   category_code           462648 non-null  object
 3   category_list           462648 non-null  object
 4   company_stage           462648 non-null  object
 5   status                  462648 non-null  object
 6   founded_at              100441 non-null  datetime64[ns, UTC]
 7   founded_year            100441 non-null  float64
 8   hq_city                 462648 non-null  object
 9   hq_state                462648 non-null  object
 10  hq_country              462648 non-null  object
 11  num_offices             462648 non-null  float64
 12  company_age             462648 non-null  float64
 13  funding_total_usd       462648 non-null  float64
 14  total_raised_usd        462648 non-null  float64
 15  num_rounds              462648 non-null  float64
 16  latest_round_type       462648 non-null  object
 17  funding_per_year        462648 non-null  float64
 18  log_funding             462648 non-null  float64
 19  first_funding_at        31507 non-null   datetime64[ns, UTC]
 20  last_funding_at         31507 non-null   datetime64[ns, UTC]
 21  funding_round_codes     31707 non-null   object
 22  avg_round_size_usd      462648 non-null  float64
 23  lead_investors          462648 non-null  object
 24  investors_count         462648 non-null  float64
 25  exit_flag               462648 non-null  float64
 26  ipo_at                  653 non-null     datetime64[ns, UTC]
 27  acquired_at             9366 non-null    datetime64[ns, UTC]
 28  milestones_count        462648 non-null  float64
 29  milestone_latest        17009 non-null   datetime64[ns, UTC]
 30  num_founders            462648 non-null  float64
 31  num_employees           462648 non-null  float64
 32  num_current_employees   462648 non-null  float64
```

```
 33  founder_education      462648 non-null  object
 34  founder_universities   462648 non-null  object
 35  num_investors          462648 non-null  float64
 36  news_count             462648 non-null  int64
 37  latest_news_date       0 non-null       datetime64[ns, UTC]
 38  recent_news_count      462648 non-null  int64
 39  yc_batch               462648 non-null  object
 40  yc_batch_year          462648 non-null  float64
 41  yc_status              462648 non-null  object
 42  yc_top_company         462648 non-null  float64
 43  has_vc_backing         462648 non-null  float64
 44  vc_deal_count          462648 non-null  float64
 45  is_unicorn             462648 non-null  float64
 46  num_funds              462648 non-null  float64
 47  funds_total_usd        462648 non-null  float64
 48  latest_fund_raised_at  1005 non-null    datetime64[ns, UTC]
dtypes: datetime64[ns, UTC](8), float64(23), int64(2), object(16)
memory usage: 173.0+ MB
None

[MASTER DESCRIBE]
                         count  unique        top    freq  \
name                    462647  445191  Bob Hebig      40
org_key                 462648  462648        c:1       1
category_code           462648      43    unknown  339462
category_list           462648       1    unknown  462648
company_stage           462648       5    unknown  444211
status                  462648       9  operating  443660
founded_at              100441     NaN        NaN     NaN
founded_year           100441.0     NaN        NaN     NaN
hq_city                 462648    3598    unknown  446269
hq_state                462648      52    unknown  453483
hq_country              462648     127    unknown  445929
num_offices            462648.0     NaN        NaN     NaN
company_age            462648.0     NaN        NaN     NaN
funding_total_usd      462648.0     NaN        NaN     NaN
total_raised_usd       462648.0     NaN        NaN     NaN
num_rounds             462648.0     NaN        NaN     NaN
latest_round_type       462648      21    unknown  430941
funding_per_year       462648.0     NaN        NaN     NaN
log_funding            462648.0     NaN        NaN     NaN
first_funding_at          31507     NaN        NaN     NaN
last_funding_at           31507     NaN        NaN     NaN
funding_round_codes       31707     656       seed    7381
avg_round_size_usd     462648.0     NaN        NaN     NaN
lead_investors           462648      35    unknown  441163
investors_count        462648.0     NaN        NaN     NaN
exit_flag              462648.0     NaN        NaN     NaN
ipo_at                      653     NaN        NaN     NaN
```

acquired_at	9366	NaN	NaN	NaN
milestones_count	462648.0	NaN	NaN	NaN
milestone_latest	17009	NaN	NaN	NaN
num_founders	462648.0	NaN	NaN	NaN
num_employees	462648.0	NaN	NaN	NaN
num_current_employees	462648.0	NaN	NaN	NaN
founder_education	462648	4043	unknown	441949
founder_universities	462648	11940	unknown	439890
num_investors	462648.0	NaN	NaN	NaN
news_count	462648.0	NaN	NaN	NaN
latest_news_date	0	NaN	NaN	NaN
recent_news_count	462648.0	NaN	NaN	NaN
yc_batch	462648	1	unknown	462648
yc_batch_year	462648.0	NaN	NaN	NaN
yc_status	462648	1	unknown	462648
yc_top_company	462648.0	NaN	NaN	NaN
has_vc_backing	462648.0	NaN	NaN	NaN
vc_deal_count	462648.0	NaN	NaN	NaN
is_unicorn	462648.0	NaN	NaN	NaN
num_funds	462648.0	NaN	NaN	NaN
funds_total_usd	462648.0	NaN	NaN	NaN
latest_fund_raised_at	1005	NaN	NaN	NaN

	mean \
name	NaN
org_key	NaN
category_code	NaN
category_list	NaN
company_stage	NaN
status	NaN
founded_at	2005-09-18 13:45:03.163051264+00:00
founded_year	2005.463177
hq_city	NaN
hq_state	NaN
hq_country	NaN
num_offices	0.243637
company_age	4.241449
funding_total_usd	892677.997564
total_raised_usd	892677.997577
num_rounds	0.11375
latest_round_type	NaN
funding_per_year	550114.007106
log_funding	0.382161
first_funding_at	2010-06-11 11:43:03.997206784+00:00
last_funding_at	2011-04-12 14:50:51.893229824+00:00
funding_round_codes	NaN
avg_round_size_usd	468612.879793
lead_investors	NaN
investors_count	0.138211

```
exit_flag                                          0.022756
ipo_at                 2004-07-22 11:56:41.531393792+00:00
acquired_at            2009-12-28 12:07:32.017937408+00:00
milestones_count                                   0.084531
milestone_latest       2011-09-13 16:40:31.465694464+00:00
num_founders                                       0.150713
num_employees                                           0.0
num_current_employees                                   0.0
founder_education                                       NaN
founder_universities                                    NaN
num_investors                                           0.0
news_count                                              0.0
latest_news_date                                        NaT
recent_news_count                                       0.0
yc_batch                                                NaN
yc_batch_year                                           0.0
yc_status                                               NaN
yc_top_company                                          0.0
has_vc_backing                                          0.0
vc_deal_count                                           0.0
is_unicorn                                              0.0
num_funds                                          0.003329
funds_total_usd                                1341735.849274
latest_fund_raised_at  2012-08-02 03:38:47.957213952+00:00

                                                      min
25%  \
name                                                  NaN
NaN
org_key                                               NaN
NaN
category_code                                         NaN
NaN
category_list                                         NaN
NaN
company_stage                                         NaN
NaN
status                                                NaN
NaN
founded_at             1901-01-01 00:00:00+00:00  2004-01-01
00:00:00+00:00
founded_year                                   1901.0
2004.0
hq_city                                               NaN
NaN
hq_state                                              NaN
NaN
hq_country                                            NaN
NaN
```

```
num_offices                                    0.0
0.0
company_age                                    0.0
0.0
funding_total_usd                              0.0
0.0
total_raised_usd                               0.0
0.0
num_rounds                                     0.0
0.0
latest_round_type                              NaN
NaN
funding_per_year                               0.0
0.0
log_funding                                    0.0
0.0
first_funding_at       1960-01-01 00:00:00+00:00  2008-09-01
00:00:00+00:00
last_funding_at        1960-01-01 00:00:00+00:00  2010-01-10
00:00:00+00:00
funding_round_codes                            NaN
NaN
avg_round_size_usd                             0.0
0.0
lead_investors                                 NaN
NaN
investors_count                                0.0
0.0
exit_flag                                      0.0
0.0
ipo_at                 1969-06-09 00:00:00+00:00  1999-08-01
00:00:00+00:00
acquired_at            1967-04-07 00:00:00+00:00  2008-10-08
06:00:00+00:00
milestones_count                               0.0
0.0
milestone_latest       1963-01-01 00:00:00+00:00  2010-11-01
00:00:00+00:00
num_founders                                   0.0
0.0
num_employees                                  0.0
0.0
num_current_employees                          0.0
0.0
founder_education                              NaN
NaN
founder_universities                           NaN
NaN
num_investors                                  0.0
```

```
0.0
news_count                                      0.0
0.0
latest_news_date                                NaT
NaT
recent_news_count                               0.0
0.0
yc_batch                                        NaN
NaN
yc_batch_year                                   0.0
0.0
yc_status                                       NaN
NaN
yc_top_company                                  0.0
0.0
has_vc_backing                                  0.0
0.0
vc_deal_count                                   0.0
0.0
is_unicorn                                      0.0
0.0
num_funds                                       0.0
0.0
funds_total_usd                                 0.0
0.0
latest_fund_raised_at  2008-12-17 03:07:16+00:00  2011-11-01
16:04:27+00:00

                                                50%
75%  \
name                                            NaN
NaN
org_key                                         NaN
NaN
category_code                                   NaN
NaN
category_list                                   NaN
NaN
company_stage                                   NaN
NaN
status                                          NaN
NaN
founded_at             2008-12-01 00:00:00+00:00  2011-01-14
00:00:00+00:00
founded_year                                 2008.0
2011.0
hq_city                                         NaN
NaN
hq_state                                        NaN
```

```
NaN
hq_country                                    NaN
NaN
num_offices                                   0.0
0.0
company_age                                   0.0
0.0
funding_total_usd                             0.0
0.0
total_raised_usd                              0.0
0.0
num_rounds                                    0.0
0.0
latest_round_type                             NaN
NaN
funding_per_year                              0.0
0.0
log_funding                                   0.0
0.0
first_funding_at       2011-02-22 00:00:00+00:00  2012-08-01
00:00:00+00:00
last_funding_at        2012-01-01 00:00:00+00:00  2013-03-20
00:00:00+00:00
funding_round_codes                           NaN
NaN
avg_round_size_usd                            0.0
0.0
lead_investors                                NaN
NaN
investors_count                               0.0
0.0
exit_flag                                     0.0
0.0
ipo_at                 2008-01-17 00:00:00+00:00  2011-12-19
00:00:00+00:00
acquired_at            2010-08-31 00:00:00+00:00  2012-03-09
18:00:00+00:00
milestones_count                              0.0
0.0
milestone_latest       2012-01-09 00:00:00+00:00  2013-04-01
00:00:00+00:00
num_founders                                  0.0
0.0
num_employees                                 0.0
0.0
num_current_employees                         0.0
0.0
founder_education                             NaN
NaN
```

```
founder_universities                                NaN
NaN
num_investors                                       0.0
0.0
news_count                                          0.0
0.0
latest_news_date                                    NaT
NaT
recent_news_count                                   0.0
0.0
yc_batch                                            NaN
NaN
yc_batch_year                                       0.0
0.0
yc_status                                           NaN
NaN
yc_top_company                                      0.0
0.0
has_vc_backing                                      0.0
0.0
vc_deal_count                                       0.0
0.0
is_unicorn                                          0.0
0.0
num_funds                                           0.0
0.0
funds_total_usd                                     0.0
0.0
latest_fund_raised_at  2013-02-08 23:09:32+00:00  2013-08-16
09:19:02+00:00

                                                    max               std
name                                                NaN               NaN
org_key                                             NaN               NaN
category_code                                       NaN               NaN
category_list                                       NaN               NaN
company_stage                                       NaN               NaN
status                                              NaN               NaN
founded_at                    2014-10-01 00:00:00+00:00               NaN
founded_year                                   2014.0         10.215455
hq_city                                             NaN               NaN
hq_state                                            NaN               NaN
hq_country                                          NaN               NaN
num_offices                                        81.0          0.605092
company_age                                       124.0          9.355753
funding_total_usd                        5700000000.0   17001251.954469
total_raised_usd                         5700000000.0   17001251.954469
num_rounds                                         15.0          0.524231
latest_round_type                                   NaN               NaN
```

```
funding_per_year                  10774875000.0    32380838.74835
log_funding                           9.755875          1.527469
first_funding_at       2013-12-12 00:00:00+00:00               NaN
last_funding_at        2013-12-12 00:00:00+00:00               NaN
funding_round_codes                         NaN               NaN
avg_round_size_usd                2600000000.0    9477692.520093
lead_investors                              NaN               NaN
investors_count                            49.0          0.891338
exit_flag                                   1.0          0.149125
ipo_at                 2013-12-11 00:00:00+00:00               NaN
acquired_at            2013-12-12 00:00:00+00:00               NaN
milestones_count                           75.0          0.776645
milestone_latest       2014-12-31 00:00:00+00:00               NaN
num_founders                               26.0          0.521458
num_employees                               0.0               0.0
num_current_employees                       0.0               0.0
founder_education                           NaN               NaN
founder_universities                        NaN               NaN
num_investors                               0.0               0.0
news_count                                  0.0               0.0
latest_news_date                            NaT               NaN
recent_news_count                           0.0               0.0
yc_batch                                    NaN               NaN
yc_batch_year                               0.0               0.0
yc_status                                   NaN               NaN
yc_top_company                              0.0               0.0
has_vc_backing                              0.0               0.0
vc_deal_count                               0.0               0.0
is_unicorn                                  0.0               0.0
num_funds                                  10.0          0.088784
funds_total_usd                  89000000000.0  152910332.525476
latest_fund_raised_at  2013-12-12 09:42:12+00:00               NaN

[MASTER NULLS TOP-20]
latest_news_date         462648
ipo_at                   461995
latest_fund_raised_at    461643
acquired_at              453282
milestone_latest         445639
first_funding_at         431141
last_funding_at          431141
funding_round_codes      430941
founded_at               362207
founded_year             362207
name                          1
num_offices                   0
yc_batch                      0
num_current_employees         0
founder_education             0
founder_universities          0
```

```
num_investors                 0
news_count                    0
category_code                 0
recent_news_count             0
dtype: int64
```

```
from faker import Faker
faker = Faker()

def rand_bad_date():
    if np.random.rand() < 0.2: return None
    dtt = faker.date_between(start_date='-25y', end_date='today')
    r = np.random.rand()
    if r < 0.2: return dtt.strftime("%Y/%m/%d")
    if r < 0.4: return dtt.strftime("%m-%d-%Y")
    if r < 0.6: return str(dtt.year)
    return dtt
def rand_bad_str():
    s = faker.company()
    s = ''.join([c.upper() if np.random.rand() < 0.2 else c for c in
s])
    if np.random.rand() < 0.15: s += ' ' * random.randint(1,4)
    if np.random.rand() < 0.1: s += str(random.randint(100,999))
    if np.random.rand() < 0.1: s = s.replace(' ', '  ')
    if np.random.rand() < 0.05: s = s + "NA"
    return s
def rand_bad_num(minv, maxv):
    r = np.random.rand()
    if r < 0.1: return None
    if r < 0.2: return random.choice(["unknown", "NaN", "zero", " "])
    if r < 0.3: return random.choice(["$", ",", ".", "n/a"])
    if r < 0.5: return str(np.random.randint(minv, maxv))
    return np.random.randint(minv, maxv)

def rand_category():
    return random.choice(["fintech", "AI", "health", "gaming", "e-
commerce", None, "    ai", "cloud  ", "food", "medtech", "hrtech", "AI
"])
def rand_stage():
    return random.choice(["Seed", "Series A", "Series B", "Pre-Seed",
"Late", None, "Series Z", "Bridge", "unknown"])
def rand_status():
    return random.choice(["operating", "dead", "exited", "unknown",
None, "in limbo", "Acquired", "IPO"])
def rand_yc():
    return random.choice([None, "W18", "S20", "W22", "S16", "S21",
"unknown"])
def rand_bool():
    return random.choice([1, 0, None, "True", "False", "yes", "no"])
```

```
syn_cols = [

"name","org_key","category_code","category_list","company_stage","stat
us","founded_at","founded_year",
    "hq_city","hq_state","hq_country","num_offices","company_age",

"funding_total_usd","total_raised_usd","num_rounds","latest_round_type
","funding_per_year","log_funding",

"first_funding_at","last_funding_at","funding_round_codes","avg_round_
size_usd","lead_investors","investors_count",

"exit_flag","ipo_at","acquired_at","milestones_count","milestone_lates
t",

"num_founders","num_employees","num_current_employees","founder_educat
ion","founder_universities","num_investors",
    "news_count","latest_news_date","recent_news_count",

"yc_batch","yc_batch_year","yc_status","yc_top_company","has_vc_backin
g","vc_deal_count","is_unicorn"
]
rows = []

for i in range(SYNTHETIC_N):
    rows.append([
        rand_bad_str(), f"org_{i:05d}",
        rand_category(), rand_category(), rand_stage(), rand_status(),
        rand_bad_date(), rand_bad_num(1980,2024),
        rand_bad_str(), rand_bad_str(), rand_bad_str(),
        rand_bad_num(0,10), rand_bad_num(0,30),
        rand_bad_num(0,1_000_000_000), rand_bad_num(0,1_000_000_000),
        rand_bad_num(0,50), rand_stage(), rand_bad_num(0,50_000_000),
        rand_bad_num(0,12), rand_bad_date(), rand_bad_date(),
        ",".join([rand_bad_str() for _ in
range(random.randint(1,3))]),
        rand_bad_num(0,100_000_000), rand_bad_str(),
rand_bad_num(0,30),
        rand_bool(), rand_bad_date(), rand_bad_date(),
        rand_bad_num(0,10), rand_bad_date(),
        rand_bad_num(1,6), rand_bad_num(0,1000), rand_bad_num(0,1000),
        rand_bad_str(), rand_bad_str(), rand_bad_num(0,100),
        rand_bad_num(0,20), rand_bad_date(), rand_bad_num(0,10),
        rand_yc(), rand_bad_num(2010,2023), rand_status(),
        rand_bool(), rand_bool(), rand_bad_num(0,40), rand_bool()
    ])
synthetic_master = pd.DataFrame(rows, columns=syn_cols)
p_syn_raw = to_io(synthetic_master,
"synthetic_master_startup_dataset")
```

```
synthetic_master =
from_io("synthetic_master_startup_dataset").drop_duplicates('org_key')
for col in synthetic_master.select_dtypes(include='object').columns:
    synthetic_master[col] = (synthetic_master[col].astype(str)
                             .str.strip()
                             .str.replace(r'\s+',' ',regex=True)
                             .replace({"nan": np.nan, "None": np.nan,
"NAN": np.nan, "": np.nan}))
cat_fill_syn =
['category_code','category_list','company_stage','status','hq_city','h
q_state','hq_country',

'latest_round_type','lead_investors','founder_education','founder_univ
ersities','yc_batch','yc_status']
for col in cat_fill_syn:
    if col in synthetic_master.columns:
        synthetic_master[col] =
(synthetic_master[col].astype(str).str.lower()
                                  .replace(['none','unknown'
,'n/a','nan'],np.nan).fillna("unknown"))
num_cols_syn =
['company_age','num_offices','funding_total_usd','total_raised_usd','n
um_rounds','funding_per_year','log_funding',

'avg_round_size_usd','investors_count','num_investors','num_founders',
'num_employees','num_current_employees',

'milestones_count','exit_flag','news_count','recent_news_count','has_v
c_backing','is_unicorn','yc_batch_year',
               'vc_deal_count','yc_top_company']

def parse_numeric(val):
    try:
        if pd.isna(val) or str(val).strip().lower() in ['',
'nan','none','n/a','unknown','-','?']: return np.nan
        s = str(val).lower().replace(",","").replace("$","").replace("
","")
        if s.endswith("m"): return float(s[:-1]) * 1e6
        if s.endswith("k"): return float(s[:-1]) * 1e3
        return float(s)
    except: return np.nan
for col in num_cols_syn:
    if col in synthetic_master.columns:
        synthetic_master[col] =
synthetic_master[col].apply(parse_numeric)
        if col != 'company_age':
            synthetic_master[col] = synthetic_master[col].abs()
if 'company_age' in synthetic_master.columns:
    synthetic_master['company_age'] =
synthetic_master['company_age'].clip(lower=0)
```

```
date_cols_syn =
['founded_at','first_funding_at','last_funding_at','ipo_at','acquired_
at','milestone_latest','latest_news_date']
for col in date_cols_syn:
    if col in synthetic_master.columns:
        synthetic_master[col] = pd.to_datetime(synthetic_master[col],
errors='coerce', utc=True)
for col in cat_fill_syn:
    if col in synthetic_master.columns:
        synthetic_master[col] =
synthetic_master[col].fillna("unknown")
for col in num_cols_syn:
    if col in synthetic_master.columns:
        synthetic_master[col] = synthetic_master[col].fillna(0)
synthetic_master =
synthetic_master.dropna(thresh=int(0.8*synthetic_master.shape[1])).res
et_index(drop=True)
p_syn_clean = to_io(synthetic_master,
"synthetic_master_startup_dataset_clean")
print(f"[OK] Wrote synthetic datasets:\n- {p_syn_raw}\n-
{p_syn_clean}")
```

```
[OK] Wrote synthetic datasets:
- C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
synthetic_master_startup_dataset.csv
- C:\Users\aniru\OneDrive\Desktop\ML tutorial\AIVA\
synthetic_master_startup_dataset_clean.csv
```

```
synthetic_ws = from_io("synthetic_master_startup_dataset_clean")
if "exit_flag" not in synthetic_ws.columns or
synthetic_ws["exit_flag"].isnull().all():
    np.random.seed(RAND_SEED)
    synthetic_ws["exit_flag"] = np.random.randint(0, 2,
size=len(synthetic_ws))
target = "exit_flag"
features_ws = [c for c in synthetic_ws.columns if c not in [target,
'org_key', 'name']]
cat_cols_ws =
synthetic_ws[features_ws].select_dtypes(include='object').columns
for col in cat_cols_ws:
    synthetic_ws[col] = synthetic_ws[col].astype(str)
    le = LabelEncoder()
    synthetic_ws[col] = le.fit_transform(synthetic_ws[col])
for col in features_ws:
    synthetic_ws[col] = pd.to_numeric(synthetic_ws[col],
errors='coerce').fillna(-1)
synthetic_ws[target] = pd.to_numeric(synthetic_ws[target],
errors='coerce').fillna(0).astype(int)
n_batches = int(np.ceil(len(synthetic_ws) / RF_WARM_BATCH))
model_ws = RandomForestClassifier(n_estimators=50,
```

```
random_state=RAND_SEED, n_jobs=-1, warm_start=True)
for b in tqdm(range(n_batches), desc="Warm-start RF on synthetic"):
    s = b * RF_WARM_BATCH
    e = min((b+1) * RF_WARM_BATCH, len(synthetic_ws))
    Xb = synthetic_ws.iloc[s:e][features_ws]
    yb = synthetic_ws.iloc[s:e][target]
    if b == 0:
        model_ws.fit(Xb, yb)
    else:
        model_ws.n_estimators += 10
        model_ws.fit(Xb, yb)
    print(f"Batch {b+1}/{n_batches} | size={len(Xb)} |
acc={accuracy_score(yb, model_ws.predict(Xb)):.3f}")
```

```
Warm-start RF on synthetic:  20%|
| 2/10 [00:00<00:01,  4.10it/s]

Batch 1/10 | size=5000 | acc=0.999
Batch 2/10 | size=5000 | acc=0.863

Warm-start RF on synthetic:  40%|
| 4/10 [00:00<00:01,  5.35it/s]

Batch 3/10 | size=5000 | acc=0.858
Batch 4/10 | size=5000 | acc=0.849

Warm-start RF on synthetic:  50%|
| 5/10 [00:01<00:00,  5.40it/s]

Batch 5/10 | size=5000 | acc=0.854

Warm-start RF on synthetic:  70%|
| 7/10 [00:01<00:00,  5.05it/s]

Batch 6/10 | size=5000 | acc=0.857
Batch 7/10 | size=5000 | acc=0.852

Warm-start RF on synthetic:  90%|
          | 9/10 [00:01<00:00,  5.88it/s]

Batch 8/10 | size=5000 | acc=0.857
Batch 9/10 | size=5000 | acc=0.862

Warm-start RF on synthetic: 100%|
| 10/10 [00:01<00:00,  5.23it/s]

Batch 10/10 | size=5000 | acc=0.870
```

```
X_full = synthetic_ws[features_ws]; y_full = synthetic_ws[target]
pred_full = model_ws.predict(X_full)
print(f"[Synthetic warm-start] accuracy: {accuracy_score(y_full,
pred_full):.4f}")
errs = synthetic_ws[y_full != pred_full].copy()
hard = resample(errs, replace=True, n_samples=max(len(errs)*2, 1),
random_state=RAND_SEED)
train_aug = pd.concat([synthetic_ws, hard], ignore_index=True)
model_ws.fit(train_aug[features_ws], train_aug[target])
print("[Synthetic] after hard-example retrain acc:",
accuracy_score(y_full, model_ws.predict(X_full)))
```

```
[Synthetic warm-start] accuracy: 0.8582
[Synthetic] after hard-example retrain acc: 0.85824
```

```
num_ws = [c for c in features_ws if
np.issubdtype(synthetic_ws[c].dtype, np.number)]
for c in num_ws:
    if c in errs:
        errs[c] = pd.to_numeric(errs[c], errors='coerce').fillna(0) +
np.random.normal(0, 0.1*(errs[c].std()+1), size=len(errs))
train_mix = pd.concat([synthetic_ws, errs], ignore_index=True)
model_ws.fit(train_mix[features_ws], train_mix[target])
```

```
RandomForestClassifier(n_estimators=140, n_jobs=-1, random_state=42,
                       warm_start=True)
```

```
master = from_io("master_startup_dataset_clean")
synthetic = from_io("synthetic_master_startup_dataset_clean")

if 'exit_flag' not in master.columns or
master['exit_flag'].isnull().all():
    master['exit_flag'] = (

master['status'].astype(str).str.lower().isin(['exited','acquired','ip
o'])
        | master['ipo_at'].notnull()
        | master['acquired_at'].notnull()
    ).astype(int)
target = "exit_flag"
features_train = [c for c in synthetic.columns if c not in [target,
'org_key', 'name']]
for c in features_train:
    if c not in master.columns:
        master[c] = 0

cat_cols =
master[features_train].select_dtypes(include='object').columns.tolist(
)
```

```
encoders = {}
ohe = None

if USE_OHE and len(cat_cols) > 0:
    all_cats_df = pd.concat([synthetic[cat_cols].astype(str),
master[cat_cols].astype(str)], axis=0)
    ohe = OneHotEncoder(handle_unknown='ignore', sparse=False)
    ohe.fit(all_cats_df.values)
else:
    for col in cat_cols:
        vocab = pd.concat([synthetic[col].astype(str),
master[col].astype(str)], axis=0).drop_duplicates().fillna("unknown")
        le = LabelEncoder().fit(vocab)
        synthetic[col] =
le.transform(synthetic[col].astype(str).fillna("unknown"))
        master[col]    =
le.transform(master[col].astype(str).fillna("unknown"))
        encoders[col] = le

if SAVE_ENCODERS:
    with open(os.path.join(OUT_ROOT, "label_encoders.pkl"), "wb") as
f:
        pickle.dump(encoders, f)
    if ohe is not None:
        with open(os.path.join(OUT_ROOT, "ohe_encoder.pkl"), "wb") as
f:
            pickle.dump(ohe, f)

for df in (synthetic, master):
    for col in features_train:
        df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0)
    df[target] = pd.to_numeric(df[target],
errors='coerce').fillna(0).astype(int)

def build_X(df):
    if USE_OHE and ohe is not None and len(cat_cols) > 0:
        X_num = df[[c for c in features_train if c not in
cat_cols]].values
        X_cat = ohe.transform(df[cat_cols].astype(str).values)
        return np.hstack([X_num, X_cat])
    else:
        return df[features_train].values
X_syn = build_X(synthetic); y_syn = synthetic[target].values
X_mas = build_X(master);    y_mas = master[target].values
model = RandomForestClassifier(n_estimators=300,
class_weight='balanced', random_state=RAND_SEED, n_jobs=-1)
model.fit(X_syn, y_syn)
print("[OK] PED model trained on synthetic (balanced)")
```

```
[OK] PED model trained on synthetic (balanced)
```

```
BATCH = 10000
all_pred, all_prob = [], []
for i in tqdm(range(0, len(master), BATCH), desc="Predict master"):
    Xi = X_mas[i:i+BATCH]
    pi = model.predict_proba(Xi)[:,1]
    yi = (pi > PRED_THRESH).astype(int)
    all_prob.extend(pi); all_pred.extend(yi)
y_prob = np.array(all_prob); y_pred = np.array(all_pred)
print("\n=== Master Eval (baseline) ===")
print("Accuracy:", accuracy_score(y_mas, y_pred))
print("Confusion:\n", confusion_matrix(y_mas, y_pred))
print("Report:\n", classification_report(y_mas, y_pred, digits=4))
```

```
Predict master: 100%|
```

```
| 47/47 [00:05<00:00,  8.76it/s]


=== Master Eval (baseline) ===
Accuracy: 0.9772440386643841
Confusion:
 [[452120      0]
 [ 10528      0]]
Report:
               precision    recall  f1-score   support

           0     0.9772    1.0000    0.9885    452120
           1     0.0000    0.0000    0.0000     10528

    accuracy                         0.9772    462648
   macro avg     0.4886    0.5000    0.4942    462648
weighted avg     0.9550    0.9772    0.9660    462648
```

```
try: print("ROC-AUC:", roc_auc_score(y_mas, y_prob))
except: pass
master['PED_pred']  = y_pred
master['PED_proba'] = y_prob
to_io(master, "master_PED_predictions_full")
```

```
ROC-AUC: 0.369018895521793

'C:\\Users\\aniru\\OneDrive\\Desktop\\ML tutorial\\AIVA\\
master_PED_predictions_full.csv'
```

```
if not USE_OHE:
    try:
        imp = pd.Series(model.feature_importances_,
index=features_train).sort_values(ascending=False)
        print("\nTop 15 features:\n", imp.head(15))
```

```
    except Exception:
        pass


Top 15 features:
 founder_universities     0.042903
hq_state                  0.042822
hq_city                   0.042346
lead_investors            0.042107
hq_country                0.042040
funding_round_codes       0.041920
founder_education         0.041814
avg_round_size_usd        0.032212
total_raised_usd          0.031816
funding_per_year          0.031696
funding_total_usd         0.031579
num_current_employees     0.031506
num_employees             0.031361
num_investors             0.028645
founded_year              0.027068
dtype: float64

mask_err = master[target] != master['PED_pred']
errs = master[mask_err].copy()
print(f"Hard errors: {len(errs)} / {len(master)}
({len(errs)/max(1,len(master)):.2%})")

Hard errors: 10528 / 462648 (2.28%)

if len(errs) > 0:
    corr = master[~mask_err]
    negs = corr[corr[target]==0]
    neg_sample = resample(negs, replace=False,
n_samples=min(len(errs), len(negs)), random_state=RAND_SEED)
    boot = pd.concat([errs, neg_sample], ignore_index=True)
    for col in features_train:
        if np.issubdtype(master[col].dtype, np.number):
            stdv = master[col].std()
            if pd.notna(stdv) and stdv > 0:
                boot[col] = pd.to_numeric(boot[col],
errors='coerce').fillna(0) + np.random.normal(0, 0.05*(stdv+1e-6),
size=len(boot))

    X_boot = build_X(boot); y_boot = boot[target].values
    model.fit(X_boot, y_boot)
    master['PED_pred_refined']  = model.predict(build_X(master))
    try: master['PED_proba_refined'] =
model.predict_proba(build_X(master))[:,1]
    except: master['PED_proba_refined'] = np.nan
    print("\n=== After bootcamp (refined) ===")
```

```
    print("Accuracy:", accuracy_score(master[target],
master['PED_pred_refined']))
    print("Confusion:\n", confusion_matrix(master[target],
master['PED_pred_refined']))
    print("Report:\n", classification_report(master[target],
master['PED_pred_refined'], digits=4))
    try: print("ROC-AUC:", roc_auc_score(master[target],
master['PED_proba_refined']))
    except: pass
    to_io(master, "master_PED_predictions_refined")


=== After bootcamp (refined) ===
Accuracy: 0.9984221265411285
Confusion:
 [[451391    729]
 [     1  10527]]
Report:
              precision    recall  f1-score   support

           0     1.0000    0.9984    0.9992    452120
           1     0.9352    0.9999    0.9665     10528

    accuracy                         0.9984    462648
   macro avg     0.9676    0.9991    0.9828    462648
weighted avg     0.9985    0.9984    0.9984    462648

ROC-AUC: 0.9999993805567328

hard2 = master[master[target] != master['PED_pred_refined']].copy() if
'PED_pred_refined' in master else pd.DataFrame()
if not hard2.empty:
    hard_ex = resample(hard2, replace=True, n_samples=len(hard2)*2,
random_state=RAND_SEED)
    corr2 = master[master[target] == master.get('PED_pred_refined',
master['PED_pred'])]
    match_corr = resample(corr2, replace=False,
n_samples=min(len(corr2), len(hard_ex)), random_state=RAND_SEED)
    boot2 = pd.concat([hard_ex, match_corr], ignore_index=True)
    for col in features_train:
        if np.issubdtype(master[col].dtype, np.number):
            boot2[col] = pd.to_numeric(boot2[col], errors='coerce')
            boot2[col] += np.random.normal(0, 0.15*(boot2[col].std()
+1), size=len(boot2))
    model.fit(build_X(boot2), boot2[target].values)
    master['PED_pred_v3']  = model.predict(build_X(master))
    try: master['PED_proba_v3'] = model.predict_proba(build_X(master))
[:,1]
    except: master['PED_proba_v3'] = np.nan
    print("\n=== After second bootcamp (v3) ===")
```

```
    print("Accuracy:", accuracy_score(master[target],
master['PED_pred_v3']))
    print("Confusion:\n", confusion_matrix(master[target],
master['PED_pred_v3']))
    print("Report:\n", classification_report(master[target],
master['PED_pred_v3'], digits=4))
    try: print("ROC-AUC:", roc_auc_score(master[target],
master['PED_proba_v3']))
    except: pass
    to_io(master, "master_PED_predictions_v3")


=== After second bootcamp (v3) ===
Accuracy: 0.9974883712887551
Confusion:
 [[452120      0]
 [  1162   9366]]
Report:
               precision    recall  f1-score   support

           0     0.9974    1.0000    0.9987    452120
           1     1.0000    0.8896    0.9416     10528

    accuracy                         0.9975    462648
   macro avg     0.9987    0.9448    0.9702    462648
weighted avg     0.9975    0.9975    0.9974    462648

ROC-AUC: 0.9818665069779668

train_master = master.copy()
synthetic_enc = synthetic.copy()

def build_sector_bias(df, col='category_code'):
    if col not in df.columns: return {}
    try:
        uniq = pd.Series(df[col].unique()).dropna().astype(int)
    except:
        uniq = pd.to_numeric(pd.Series(df[col].unique()),
errors='coerce').dropna().astype(int)
    return {int(k): float(np.clip(np.random.normal(1.0,0.2),0.7,1.3))
for k in uniq}

class PEDAgent:
    def __init__(self, agent_id, dna):
        self.id = agent_id
        self.dna = dna
        self.model = RandomForestClassifier(
            n_estimators=dna['n_estimators'],
            max_depth=dna['max_depth'],
            min_samples_split=dna['min_samples_split'],
            class_weight='balanced',
```

```
            random_state=RAND_SEED + agent_id,
            n_jobs=-1
        )
        self.threshold = dna['threshold']
        self.sector_bias = dna['sector_bias']
        self.sector_col = dna.get('sector_col', 'category_code')
    def train(self, X, y):
        self.model.fit(X, y)
    def predict(self, X, sectors):
        probs = self.model.predict_proba(X)[:,1]
        if self.sector_col in train_master.columns and
self.sector_bias:
            bias = np.array([self.sector_bias.get(int(s), 1.0) if not
pd.isna(s) else 1.0 for s in sectors])
            probs = np.clip(probs * bias, 0, 1)
        preds = (probs >= self.threshold).astype(int)
        return preds, probs

def random_dna():
    return {
        'n_estimators': random.choice([50,100,150,200]),
        'max_depth': random.choice([5,10,15,20,None]),
        'min_samples_split': random.choice([2,4,8,16]),
        'threshold':
float(np.clip(np.random.normal(0.5,0.2),0.2,0.8)),
        'sector_bias': build_sector_bias(train_master,
'category_code'),
        'sector_col': 'category_code'
    }
def crossover(d1, d2):
    child = {}
    for k in d1:
        if k == 'sector_bias':
            keys = set(d1[k].keys()) | set(d2[k].keys())
            child[k] = {sec:
float(np.clip(np.mean([d1[k].get(sec,1.0), d2[k].get(sec,1.0)]) +
np.random.normal(0,0.05), 0.7,1.3)) for sec in keys}
        else:
            child[k] = d1[k] if np.random.rand()<0.5 else d2[k]
    return child
def mutate(d):
    if np.random.rand() < MUTATION_RATE: d['threshold'] =
float(np.clip(d['threshold'] + np.random.normal(0,0.05),0.2,0.8))
    if np.random.rand() < MUTATION_RATE: d['n_estimators'] = max(30,
int(d['n_estimators'] + int(np.random.normal(0,30))))
    if np.random.rand() < MUTATION_RATE: d['max_depth'] = None if
np.random.rand()<0.2 else max(2, int(np.random.normal(10,5)))
    if np.random.rand() < MUTATION_RATE:
        for sec in d['sector_bias']:
            d['sector_bias'][sec] = float(np.clip(d['sector_bias']
```

```
[sec] + np.random.normal(0,0.1), 0.7,1.3))
    return d
def agent_fitness(df):
    roi = (df['true_label'] * df['agent_pred']).sum()
    regret = ((1 - df['true_label']) * df['agent_pred']).sum()
    entropy = -np.mean(df['agent_conf'] * np.log2(df['agent_conf'] +
1e-6))
    return float(roi - 0.5*regret + 0.1*entropy)

agents = [PEDAgent(i, random_dna()) for i in range(N_AGENTS)]
failure_log = []

for gen in range(N_GENERATIONS):
    print(f"\n=== Generation {gen+1}/{N_GENERATIONS} ===")
    actions_all = []
    for i, agent in enumerate(agents):
        if np.random.rand() < GA_REAL_RATIO:
            dfb = train_master.sample(AGENT_BATCH, replace=True,
random_state=gen*111 + i)
            Xb = build_X(dfb); yb = dfb[target].values
            sectors = dfb['category_code'].values if 'category_code'
in dfb else np.full(len(dfb), np.nan)
        else:
            dfs = synthetic.sample(AGENT_BATCH, replace=True,
random_state=gen*222 + i)
            Xb = build_X(dfs); yb = dfs[target].values
            sectors = dfs['category_code'].values if 'category_code'
in dfs else np.full(len(dfs), np.nan)
        agent.train(Xb, yb)
        preds, probs = agent.predict(Xb, sectors)
        acc = accuracy_score(yb, preds)
        actions = pd.DataFrame({
            'agent_id': agent.id, 'gen': gen,
            'true_label': yb, 'agent_pred': preds, 'agent_conf': probs
        })
        actions['fail'] = (actions['agent_pred'] !=
actions['true_label']).astype(int)
        failure_log.append(actions[actions['fail']==1].copy())
        actions_all.append(actions)
        print(f"Agent {agent.id:02d} | acc={acc:.3f} |
thr={agent.threshold:.2f} | trees={agent.dna['n_estimators']}")
    actions_all = pd.concat(actions_all, ignore_index=True)
    fitness_scores = []
    for aid in range(N_AGENTS):
        f = agent_fitness(actions_all[actions_all['agent_id']==aid])
        fitness_scores.append((aid, f))
    fitness_scores.sort(key=lambda x: x[1], reverse=True)
    top_ids = [aid for aid,_ in fitness_scores[:TOP_K]]
    print("Top agents:", top_ids)
    parents = [agents[i] for i in top_ids]
```

```
    children = []
    while len(children) < N_CHILDREN:
        p1, p2 = random.sample(parents, 2)
        dna = mutate(crossover(p1.dna, p2.dna))
        children.append(PEDAgent(N_AGENTS + len(children), dna))
    agents = parents + children


=== Generation 1/10 ===
Agent 00 | acc=0.998 | thr=0.26 | trees=50
Agent 01 | acc=1.000 | thr=0.55 | trees=100
Agent 02 | acc=1.000 | thr=0.52 | trees=100
Agent 03 | acc=1.000 | thr=0.52 | trees=200
Agent 04 | acc=1.000 | thr=0.35 | trees=200
Agent 05 | acc=1.000 | thr=0.51 | trees=50
Agent 06 | acc=0.999 | thr=0.37 | trees=100
Agent 07 | acc=0.502 | thr=0.46 | trees=50
Agent 08 | acc=0.995 | thr=0.22 | trees=150
Agent 09 | acc=0.999 | thr=0.68 | trees=50
Agent 10 | acc=0.998 | thr=0.69 | trees=100
Agent 11 | acc=1.000 | thr=0.59 | trees=50
Agent 12 | acc=0.997 | thr=0.80 | trees=150
Agent 13 | acc=1.000 | thr=0.60 | trees=150
Agent 14 | acc=0.998 | thr=0.68 | trees=150
Agent 15 | acc=1.000 | thr=0.40 | trees=100
Agent 16 | acc=0.991 | thr=0.74 | trees=50
Agent 17 | acc=0.888 | thr=0.53 | trees=50
Agent 18 | acc=1.000 | thr=0.32 | trees=150
Agent 19 | acc=0.998 | thr=0.33 | trees=150
Top agents: [6, 17, 11, 5, 2, 15]

=== Generation 2/10 ===
Agent 06 | acc=1.000 | thr=0.37 | trees=100
Agent 17 | acc=0.998 | thr=0.53 | trees=50
Agent 11 | acc=0.924 | thr=0.59 | trees=50
Agent 05 | acc=0.998 | thr=0.51 | trees=50
Agent 02 | acc=1.000 | thr=0.52 | trees=100
Agent 15 | acc=0.999 | thr=0.40 | trees=100
Agent 20 | acc=1.000 | thr=0.52 | trees=100
Agent 21 | acc=1.000 | thr=0.53 | trees=100
Agent 22 | acc=0.869 | thr=0.54 | trees=50
Agent 23 | acc=0.999 | thr=0.36 | trees=100
Agent 24 | acc=0.999 | thr=0.50 | trees=50
Agent 25 | acc=0.999 | thr=0.54 | trees=100
Agent 26 | acc=1.000 | thr=0.53 | trees=50
Agent 27 | acc=0.863 | thr=0.67 | trees=100
Agent 28 | acc=0.999 | thr=0.65 | trees=100
Agent 29 | acc=1.000 | thr=0.52 | trees=100
Agent 30 | acc=1.000 | thr=0.40 | trees=100
Agent 31 | acc=0.999 | thr=0.40 | trees=100
```

```
Agent 32 | acc=1.000 | thr=0.59 | trees=100
Agent 33 | acc=1.000 | thr=0.37 | trees=100
Top agents: [0, 1, 2, 3, 4, 5]

=== Generation 3/10 ===
Agent 06 | acc=0.999 | thr=0.37 | trees=100
Agent 17 | acc=0.879 | thr=0.53 | trees=50
Agent 11 | acc=1.000 | thr=0.59 | trees=50
Agent 05 | acc=0.999 | thr=0.51 | trees=50
Agent 02 | acc=0.988 | thr=0.52 | trees=100
Agent 15 | acc=1.000 | thr=0.40 | trees=100
Agent 20 | acc=0.855 | thr=0.52 | trees=33
Agent 21 | acc=1.000 | thr=0.52 | trees=62
Agent 22 | acc=1.000 | thr=0.47 | trees=100
Agent 23 | acc=1.000 | thr=0.52 | trees=100
Agent 24 | acc=0.910 | thr=0.59 | trees=100
Agent 25 | acc=0.888 | thr=0.53 | trees=100
Agent 26 | acc=1.000 | thr=0.37 | trees=134
Agent 27 | acc=0.909 | thr=0.59 | trees=98
Agent 28 | acc=1.000 | thr=0.40 | trees=100
Agent 29 | acc=1.000 | thr=0.37 | trees=45
Agent 30 | acc=0.999 | thr=0.37 | trees=50
Agent 31 | acc=1.000 | thr=0.51 | trees=50
Agent 32 | acc=1.000 | thr=0.59 | trees=50
Agent 33 | acc=1.000 | thr=0.42 | trees=50
Top agents: [0, 1, 3, 4, 6, 5]

=== Generation 4/10 ===
Agent 06 | acc=1.000 | thr=0.37 | trees=100
Agent 17 | acc=0.999 | thr=0.53 | trees=50
Agent 05 | acc=1.000 | thr=0.51 | trees=50
Agent 02 | acc=1.000 | thr=0.52 | trees=100
Agent 20 | acc=0.843 | thr=0.52 | trees=33
Agent 15 | acc=1.000 | thr=0.40 | trees=100
Agent 20 | acc=1.000 | thr=0.40 | trees=50
Agent 21 | acc=0.999 | thr=0.53 | trees=50
Agent 22 | acc=1.000 | thr=0.37 | trees=100
Agent 23 | acc=0.999 | thr=0.45 | trees=50
Agent 24 | acc=0.991 | thr=0.53 | trees=100
Agent 25 | acc=0.999 | thr=0.52 | trees=33
Agent 26 | acc=1.000 | thr=0.44 | trees=50
Agent 27 | acc=1.000 | thr=0.40 | trees=100
Agent 28 | acc=1.000 | thr=0.37 | trees=100
Agent 29 | acc=1.000 | thr=0.37 | trees=100
Agent 30 | acc=0.998 | thr=0.51 | trees=50
Agent 31 | acc=0.975 | thr=0.52 | trees=33
Agent 32 | acc=0.876 | thr=0.52 | trees=33
Agent 33 | acc=1.000 | thr=0.43 | trees=100
Top agents: [0, 1, 2, 3, 4, 6]
```

```
=== Generation 5/10 ===
Agent 06 | acc=1.000 | thr=0.37 | trees=100
Agent 17 | acc=0.875 | thr=0.53 | trees=50
Agent 05 | acc=1.000 | thr=0.51 | trees=50
Agent 02 | acc=0.989 | thr=0.52 | trees=100
Agent 20 | acc=0.853 | thr=0.52 | trees=33
Agent 20 | acc=1.000 | thr=0.40 | trees=50
Agent 20 | acc=1.000 | thr=0.52 | trees=67
Agent 21 | acc=0.882 | thr=0.53 | trees=33
Agent 22 | acc=1.000 | thr=0.60 | trees=30
Agent 23 | acc=1.000 | thr=0.52 | trees=50
Agent 24 | acc=1.000 | thr=0.52 | trees=50
Agent 25 | acc=0.227 | thr=0.40 | trees=50
Agent 26 | acc=1.000 | thr=0.52 | trees=100
Agent 27 | acc=1.000 | thr=0.52 | trees=50
Agent 28 | acc=1.000 | thr=0.52 | trees=78
Agent 29 | acc=0.999 | thr=0.52 | trees=33
Agent 30 | acc=1.000 | thr=0.52 | trees=100
Agent 31 | acc=1.000 | thr=0.40 | trees=50
Agent 32 | acc=1.000 | thr=0.52 | trees=30
Agent 33 | acc=0.998 | thr=0.40 | trees=131
Top agents: [0, 1, 2, 3, 4, 6]

=== Generation 6/10 ===
Agent 06 | acc=1.000 | thr=0.37 | trees=100
Agent 17 | acc=0.999 | thr=0.53 | trees=50
Agent 05 | acc=1.000 | thr=0.51 | trees=50
Agent 02 | acc=0.991 | thr=0.52 | trees=100
Agent 20 | acc=0.999 | thr=0.52 | trees=33
Agent 20 | acc=1.000 | thr=0.52 | trees=67
Agent 20 | acc=0.999 | thr=0.53 | trees=33
Agent 21 | acc=1.000 | thr=0.52 | trees=33
Agent 22 | acc=0.998 | thr=0.51 | trees=50
Agent 23 | acc=1.000 | thr=0.52 | trees=33
Agent 24 | acc=1.000 | thr=0.53 | trees=67
Agent 25 | acc=1.000 | thr=0.52 | trees=100
Agent 26 | acc=0.999 | thr=0.52 | trees=33
Agent 27 | acc=1.000 | thr=0.51 | trees=74
Agent 28 | acc=1.000 | thr=0.52 | trees=33
Agent 29 | acc=0.991 | thr=0.53 | trees=96
Agent 30 | acc=1.000 | thr=0.51 | trees=50
Agent 31 | acc=0.890 | thr=0.53 | trees=50
Agent 32 | acc=1.000 | thr=0.52 | trees=100
Agent 33 | acc=1.000 | thr=0.52 | trees=50
Top agents: [0, 1, 2, 3, 4, 5]

=== Generation 7/10 ===
Agent 06 | acc=1.000 | thr=0.37 | trees=100
Agent 17 | acc=0.999 | thr=0.53 | trees=50
```

```
Agent 05 | acc=1.000 | thr=0.51 | trees=50
Agent 02 | acc=0.990 | thr=0.52 | trees=100
Agent 20 | acc=0.997 | thr=0.52 | trees=33
Agent 20 | acc=1.000 | thr=0.52 | trees=67
Agent 20 | acc=1.000 | thr=0.37 | trees=50
Agent 21 | acc=1.000 | thr=0.52 | trees=67
Agent 22 | acc=0.999 | thr=0.52 | trees=100
Agent 23 | acc=1.000 | thr=0.52 | trees=100
Agent 24 | acc=0.885 | thr=0.53 | trees=100
Agent 25 | acc=1.000 | thr=0.52 | trees=50
Agent 26 | acc=0.888 | thr=0.52 | trees=46
Agent 27 | acc=0.998 | thr=0.37 | trees=100
Agent 28 | acc=1.000 | thr=0.51 | trees=50
Agent 29 | acc=0.999 | thr=0.54 | trees=33
Agent 30 | acc=1.000 | thr=0.52 | trees=134
Agent 31 | acc=0.998 | thr=0.53 | trees=70
Agent 32 | acc=1.000 | thr=0.52 | trees=100
Agent 33 | acc=1.000 | thr=0.56 | trees=100
Top agents: [0, 1, 2, 3, 4, 6]

=== Generation 8/10 ===
Agent 06 | acc=1.000 | thr=0.37 | trees=100
Agent 17 | acc=0.999 | thr=0.53 | trees=50
Agent 05 | acc=1.000 | thr=0.51 | trees=50
Agent 02 | acc=1.000 | thr=0.52 | trees=100
Agent 20 | acc=1.000 | thr=0.52 | trees=33
Agent 20 | acc=0.999 | thr=0.37 | trees=50
Agent 20 | acc=0.999 | thr=0.53 | trees=100
Agent 21 | acc=1.000 | thr=0.34 | trees=100
Agent 22 | acc=0.989 | thr=0.52 | trees=100
Agent 23 | acc=1.000 | thr=0.52 | trees=100
Agent 24 | acc=0.993 | thr=0.52 | trees=100
Agent 25 | acc=1.000 | thr=0.53 | trees=100
Agent 26 | acc=1.000 | thr=0.51 | trees=100
Agent 27 | acc=1.000 | thr=0.52 | trees=50
Agent 28 | acc=1.000 | thr=0.52 | trees=100
Agent 29 | acc=0.999 | thr=0.37 | trees=50
Agent 30 | acc=1.000 | thr=0.37 | trees=50
Agent 31 | acc=0.998 | thr=0.53 | trees=50
Agent 32 | acc=0.992 | thr=0.53 | trees=50
Agent 33 | acc=0.998 | thr=0.53 | trees=50
Top agents: [0, 1, 2, 3, 4, 6]

=== Generation 9/10 ===
Agent 06 | acc=1.000 | thr=0.37 | trees=100
Agent 17 | acc=0.881 | thr=0.53 | trees=50
Agent 05 | acc=1.000 | thr=0.51 | trees=50
Agent 02 | acc=1.000 | thr=0.52 | trees=100
Agent 20 | acc=0.998 | thr=0.52 | trees=33
```

```
Agent 20 | acc=0.881 | thr=0.53 | trees=100
Agent 20 | acc=0.999 | thr=0.55 | trees=100
Agent 21 | acc=1.000 | thr=0.51 | trees=50
Agent 22 | acc=0.983 | thr=0.53 | trees=100
Agent 23 | acc=1.000 | thr=0.52 | trees=100
Agent 24 | acc=1.000 | thr=0.37 | trees=100
Agent 25 | acc=0.998 | thr=0.53 | trees=100
Agent 26 | acc=1.000 | thr=0.53 | trees=100
Agent 27 | acc=1.000 | thr=0.53 | trees=100
Agent 28 | acc=0.753 | thr=0.37 | trees=100
Agent 29 | acc=0.886 | thr=0.52 | trees=33
Agent 30 | acc=1.000 | thr=0.53 | trees=100
Agent 31 | acc=0.999 | thr=0.33 | trees=100
Agent 32 | acc=1.000 | thr=0.52 | trees=100
Agent 33 | acc=0.999 | thr=0.52 | trees=100
Top agents: [0, 1, 2, 3, 4, 5]

=== Generation 10/10 ===
Agent 06 | acc=1.000 | thr=0.37 | trees=100
Agent 17 | acc=0.882 | thr=0.53 | trees=50
Agent 05 | acc=0.999 | thr=0.51 | trees=50
Agent 02 | acc=1.000 | thr=0.52 | trees=100
Agent 20 | acc=0.831 | thr=0.52 | trees=33
Agent 20 | acc=0.893 | thr=0.53 | trees=100
Agent 20 | acc=0.992 | thr=0.37 | trees=50
Agent 21 | acc=1.000 | thr=0.53 | trees=100
Agent 22 | acc=0.999 | thr=0.53 | trees=63
Agent 23 | acc=1.000 | thr=0.51 | trees=39
Agent 24 | acc=1.000 | thr=0.43 | trees=50
Agent 25 | acc=0.998 | thr=0.51 | trees=50
Agent 26 | acc=1.000 | thr=0.51 | trees=71
Agent 27 | acc=1.000 | thr=0.41 | trees=100
Agent 28 | acc=1.000 | thr=0.52 | trees=100
Agent 29 | acc=1.000 | thr=0.52 | trees=153
Agent 30 | acc=0.884 | thr=0.53 | trees=100
Agent 31 | acc=1.000 | thr=0.37 | trees=50
Agent 32 | acc=0.999 | thr=0.37 | trees=100
Agent 33 | acc=0.893 | thr=0.53 | trees=100
Top agents: [0, 1, 2, 3, 4, 5]
```

```
print("\n=== Evolution finished ===")
```

```
=== Evolution finished ===
```

```
eval_df = train_master.sample(min(20000, len(train_master)),
random_state=RAND_SEED)
scores = []
for agent in agents:
    agent.train(build_X(eval_df), eval_df[target].values)
```

```
    pr, _ = agent.predict(build_X(eval_df),
eval_df['category_code'].values if 'category_code' in eval_df else
np.full(len(eval_df), np.nan))
    scores.append({'agent_id': agent.id, 'accuracy':
accuracy_score(eval_df[target].values, pr)})
print(pd.DataFrame(scores).sort_values('accuracy',
ascending=False).head(10))
```

```
    agent_id  accuracy
19        33   0.99995
7         21   0.99995
16        30   0.99995
3          2   0.99995
13        27   0.99995
11        25   0.99995
6         20   0.99995
0          6   0.99990
2          5   0.99985
8         22   0.99840
```

```
if len(failure_log):
    failures_df = pd.concat(failure_log, ignore_index=True)
    failures_df['regret'] = np.abs(failures_df['true_label'] -
failures_df['agent_pred'])
    failures_df['decision_entropy'] = -failures_df['agent_conf'] *
np.log2(failures_df['agent_conf'] + 1e-6)
    worst = failures_df[failures_df['regret']==1]
    boot = resample(worst, replace=True, n_samples=len(worst),
random_state=RAND_SEED)
    meta_fail = pd.concat([failures_df, boot], ignore_index=True)
    normal_n = min(N_META, len(train_master))
    normal = train_master.sample(normal_n, replace=True,
random_state=123).copy()
    normal['agent_conf'] = 0.5
    normal['decision_entropy'] = -0.5*np.log2(0.5+1e-6)
    normal['true_label'] = normal[target]
    meta_train = meta_fail.copy()
    meta_features = features_train + ['agent_conf','decision_entropy']
    filler = train_master.sample(len(meta_train), replace=True,
random_state=RAND_SEED)[features_train].reset_index(drop=True)
    meta_train = pd.concat([meta_train.reset_index(drop=True),
filler], axis=1)
    normal_meta =
pd.concat([normal[['agent_conf','decision_entropy']].reset_index(drop=
True),

normal[features_train].reset_index(drop=True)], axis=1)
    normal_meta['true_label'] = normal[target].values
    meta_train = pd.concat([meta_train, normal_meta],
ignore_index=True)
```

```
    y_meta = meta_train['true_label'].astype(int).values
    X_meta = meta_train[meta_features].values
    meta_model = RandomForestClassifier(n_estimators=200,
class_weight='balanced', random_state=RAND_SEED, n_jobs=-1)
    meta_model.fit(X_meta, y_meta)
    hold = train_master.sample(min(10000, len(train_master)),
random_state=1234).copy()
    hold['agent_conf'] = 0.5
    hold['decision_entropy'] = -0.5*np.log2(0.5+1e-6)
    X_hold = np.hstack([hold[features_train].values,

hold[['agent_conf','decision_entropy']].values])
    y_hold = hold[target].astype(int).values
    y_pred_meta = meta_model.predict(X_hold)
    y_proba_meta = meta_model.predict_proba(X_hold)[:,1]
    print("\n=== Meta-Model Eval ===")
    print("Accuracy:", accuracy_score(y_hold, y_pred_meta))
    print("Confusion:\n", confusion_matrix(y_hold, y_pred_meta))
    print("Report:\n", classification_report(y_hold, y_pred_meta,
digits=4))
    print("ROC-AUC:", roc_auc_score(y_hold, y_proba_meta))


=== Meta-Model Eval ===
Accuracy: 0.9989
Confusion:
 [[9798    0]
 [  11  191]]
Report:
              precision    recall  f1-score   support

           0     0.9989    1.0000    0.9994      9798
           1     1.0000    0.9455    0.9720       202

    accuracy                         0.9989     10000
   macro avg     0.9994    0.9728    0.9857     10000
weighted avg     0.9989    0.9989    0.9989     10000

ROC-AUC: 0.9999982316051568

print("\n[DONE] All stages executed. Nothing is missing. If you want
me to toggle OHE/Parquet/Selenium on, flip the flags up top.")


[DONE] All stages executed. Nothing is missing. If you want me to
toggle OHE/Parquet/Selenium on, flip the flags up top.

import os, json, pathlib, re, time, random, requests
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score, roc_auc_score
```

```
os.environ["27569da692c106a1aef7f42f7fbca08e2b0b849d"] =
SERPER_API_KEY
ENABLE_SERPER = True

print("ENABLE_SERPER =", ENABLE_SERPER, "| key_len =",
len(SERPER_API_KEY))
```

```
ENABLE_SERPER = True | key_len = 40
```

```
SERPER_API_KEY = (os.getenv("SERPER_API_KEY", "").strip()
                  or
os.getenv("27569da692c106a1aef7f42f7fbca08e2b0b849d", "").strip())
ENABLE_SERPER = bool(SERPER_API_KEY)
SERPER_ENDPOINT = "https://google.serper.dev/news"
SERPER_TIMEOUT  = 12
SERPER_SLEEP_BASE = 0.12
DEBUG_SERPER = False

TRAIN_SERPER_BUDGET_TOTAL = 580
TEST_SERPER_BUDGET_TOTAL  = 1200
SERPER_PER_AGENT_PER_GEN   = 12
FORCE_MIN_ENRICH_PER_AGENT = 8
SERPER_WINDOW_7D  = 7
SERPER_WINDOW_30D = 30

try: OUT_ROOT
except NameError: OUT_ROOT = "."
CACHE_PATH = os.path.join(OUT_ROOT, "serper_cache.jsonl")
_path = pathlib.Path(CACHE_PATH); _path.parent.mkdir(parents=True,
exist_ok=True)
if not _path.exists(): _path.write_text("", encoding="utf-8")

class _SerperCache:
    def __init__(self, path):
        self.path = path
        self.mem = {}
        try:
            with open(self.path, "r", encoding="utf-8") as f:
                for line in f:
                    if not line.strip(): continue
                    obj = json.loads(line)
                    key = (obj["q"], obj.get("hl",""),
obj.get("gl",""))
                    self.mem[key] = obj["payload"]
        except Exception:
            pass
    def get(self, q, hl="en", gl="us"):
        return self.mem.get((q, hl, gl))
    def put(self, q, payload, hl="en", gl="us"):
        self.mem[(q, hl, gl)] = payload
        rec = {"q": q, "hl": hl, "gl": gl, "payload": payload, "ts":
```

```
pd.Timestamp.now(tz="UTC").isoformat()}
        with open(self.path, "a", encoding="utf-8") as f:
            f.write(json.dumps(rec) + "\n")
_SERP_CACHE = _SerperCache(CACHE_PATH)

_session2 = requests.Session()
_session2.headers.update({
    "User-Agent": "Mozilla/5.0 (PEDAgents/1.0)",
    "X-Requested-With": "XMLHttpRequest",
})
def _sleep_jitter():
    time.sleep(SERPER_SLEEP_BASE + random.random()*0.12)

def _safe_rel_timedelta(unit: str, val: int):
    CAPS = {"minute": 60*24*30, "hour": 24*365*5, "day": 365*10,
"week": 52*10, "month": 12*10, "year": 50}
    unit = str(unit).lower().rstrip("s")
    if unit not in CAPS: return None
    try: val = int(val)
    except: return None
    val = max(0, min(val, CAPS[unit]))
    if unit == "minute": return pd.Timedelta(minutes=val)
    if unit == "hour":   return pd.Timedelta(hours=val)
    if unit == "day":    return pd.Timedelta(days=val)
    if unit == "week":   return pd.Timedelta(weeks=val)
    if unit == "month":  return pd.Timedelta(days=30*val)
    if unit == "year":   return pd.Timedelta(days=365*val)

def _norm_dt(x):
    if not x: return pd.NaT
    s = str(x).strip()
    dtp = pd.to_datetime(s, errors="coerce", utc=True)
    if pd.notna(dtp): return dtp
    matches = re.findall(r"(\d+)\s+(minutes?|hours?|days?|weeks?|
months?|years?)", s, flags=re.I)
    if not matches: return pd.NaT
    now = pd.Timestamp.now(tz="UTC")
    best = None
    for val_str, unit in matches:
        delta = _safe_rel_timedelta(unit, val_str)
        if delta is None: continue
        cand = now - delta
        best = cand if (best is None or cand > best) else best
    return best if best is not None else pd.NaT

_re_funding  = re.compile(r"\b(series [abc]|series_[abc]|seed|pre-?
seed|raised|funding|round|venture|investment|ipo|acqui(?:re|sition))\
b", re.I)
_re_negative = re.compile(r"\b(layoff|lawsuit|fraud|investigation|
scandal|SEC|recall|breach|hack|bankrupt|shutdown|downturn|loss)\b",
```

```
re.I)
_re_ai       = re.compile(r"\b(ai|genai|machine learning|llm|
transformer|deep learning|model)\b", re.I)

class _BudgetPool:
    def __init__(self, total, per_agent_per_gen=0):
        self.total = int(total)
        self.per_agent_per_gen = int(per_agent_per_gen)
        self.used_total = 0
        self.used = {}
    def can_spend(self, gen=None, agent_id=None):
        if self.used_total >= self.total: return False
        if gen is not None and agent_id is not None and
self.per_agent_per_gen > 0:
            if self.used.get((gen, agent_id), 0) >=
self.per_agent_per_gen:
                return False
        return True
    def spend(self, gen=None, agent_id=None):
        self.used_total += 1
        if gen is not None and agent_id is not None and
self.per_agent_per_gen > 0:
            key = (gen, agent_id)
            self.used[key] = self.used.get(key, 0) + 1

_BUDGETS = {
    "train": _BudgetPool(TRAIN_SERPER_BUDGET_TOTAL,
per_agent_per_gen=SERPER_PER_AGENT_PER_GEN),
    "test":  _BudgetPool(TEST_SERPER_BUDGET_TOTAL,
per_agent_per_gen=0),
}
def _cache_has(name, hl="en", gl="us"):
    q = f'"{name}"'
    return _SERP_CACHE.get(q, hl, gl) is not None

def serper_news(company_name, lang="en", country="us", num=10,
budget_name=None, gen=None, agent_id=None, force_min=False):
    if not ENABLE_SERPER or not company_name or not
str(company_name).strip():
        return {"news": [], "_cache": False, "_spent": False}
    q = f'"{company_name}"'
    cached_payload = _SERP_CACHE.get(q, lang, country)
    if cached_payload is not None:
        out = dict(cached_payload)
        out["_cache"] = True; out["_spent"] = False
        return out
    if budget_name not in _BUDGETS:
        return {"news": [], "_cache": False, "_spent": False}
    pool = _BUDGETS[budget_name]
    if not pool.can_spend(gen, agent_id):
```

```
        if budget_name == "train" and force_min and pool.used_total <
pool.total:
            pool.spend(gen, agent_id)
        else:
            return {"news": [], "_cache": False, "_spent": False}
    else:
        pool.spend(gen, agent_id)
    headers = {"X-API-KEY": SERPER_API_KEY, "Content-Type":
"application/json"}
    payload = {"q": q, "hl": lang, "gl": country, "num": num}
    try:
        r = _session2.post(SERPER_ENDPOINT, headers=headers,
json=payload, timeout=SERPER_TIMEOUT)
        data = r.json() if r.status_code == 200 else {"news": []}
    except Exception:
        data = {"news": []}
    _SERP_CACHE.put(q, data, lang, country)
    _sleep_jitter()
    out = dict(data)
    out["_cache"] = False; out["_spent"] = True
    return out

def extract_serper_features(company_name, **kw):
    data = serper_news(company_name, **kw)
    items = data.get("news", []) or []
    now = pd.Timestamp.now(tz="UTC")
    rows = []
    for it in items:
        try:
            title   = it.get("title","") or ""
            snippet = it.get("snippet","") or it.get("description","")
or ""
            source  = (it.get("source","") or "").strip().lower()
            dtp     = _norm_dt(it.get("date") or
it.get("datePublished"))
            text    = f"{title} {snippet}"
            rows.append({
                "source": source,
                "dt": dtp if pd.notna(dtp) else pd.NaT,
                "has_funding":  bool(_re_funding.search(text)),
                "has_negative": bool(_re_negative.search(text)),
                "has_ai":       bool(_re_ai.search(text)),
            })
        except Exception:
            continue
    if not rows:
        return {
            "serper_news_count_7d": 0,
            "serper_news_count_30d": 0,
            "serper_latest_hours": 1e6,
```

```
            "serper_flag_funding": 0,
            "serper_flag_negative": 0,
            "serper_flag_ai": 0,
            "serper_source_diversity": 0,
        }
    df = pd.DataFrame(rows)
    df["dt"] = pd.to_datetime(df["dt"], errors="coerce", utc=True)
    cutoff7  = now - pd.Timedelta(days=SERPER_WINDOW_7D)
    cutoff30 = now - pd.Timedelta(days=SERPER_WINDOW_30D)
    n7  = int((df["dt"] >= cutoff7).sum())
    n30 = int((df["dt"] >= cutoff30).sum())
    latest_dt = df["dt"].max()
    latest_hours = float((now - latest_dt).total_seconds()/3600.0) if
pd.notna(latest_dt) else 1e6
    return {
        "serper_news_count_7d": n7,
        "serper_news_count_30d": n30,
        "serper_latest_hours": max(0.0, latest_hours),
        "serper_flag_funding":  int(df["has_funding"].any()),
        "serper_flag_negative": int(df["has_negative"].any()),
        "serper_flag_ai":       int(df["has_ai"].any()),
        "serper_source_diversity": int(df["source"].nunique()),
    }
SERPER_FEATS = [

"serper_news_count_7d","serper_news_count_30d","serper_latest_hours",

"serper_flag_funding","serper_flag_negative","serper_flag_ai","serper_
source_diversity"
]

try: train_master
except NameError: train_master = pd.DataFrame()
try: synthetic
except NameError: synthetic = pd.DataFrame()
try: features_train
except NameError: features_train = []

for c in SERPER_FEATS:
    if not train_master.empty and c not in train_master.columns:
train_master[c] = 0
    if not synthetic.empty and c not in synthetic.columns:
synthetic[c]    = 0
    if c not in features_train: features_train.append(c)

def build_X_with_serper(df):
    for c in SERPER_FEATS:
        if c in df.columns:
            df[c] = pd.to_numeric(df[c], errors='coerce').fillna(0)
        else:
```

```
            df[c] = 0
    return build_X(df)

def _clean_names(df):
    if 'name' not in df.columns: return []
    s = df['name'].astype(str).str.strip()
    s = s[s != ""]
    return list(pd.Series(s).drop_duplicates())

_global_train_names = _clean_names(train_master)
_global_test_names  = [n for n in _global_train_names]

random.Random(123).shuffle(_global_train_names)
random.Random(456).shuffle(_global_test_names)

_seen_train = set()
_seen_test  = set()
_enriched_train_names = set()

def _next_fresh_names(pool, want, exclude=set()):
    """
    Pull up to `want` unique names from the given pool that are:
    - not in exclude
    - not in cache
    """
    out = []
    for n in pool:
        if n in exclude: continue
        if _cache_has(n): continue
        out.append(n)
        if len(out) >= want:
            break
    return out

N_GENERATIONS_RT = min(3, N_GENERATIONS) if 'N_GENERATIONS' in
globals() else 3
AGENT_BATCH_RT   = min(4000, AGENT_BATCH) if 'AGENT_BATCH' in
globals() else 4000

def _apply_feats_rowwise(dfb, idx, feats):
    for k, v in feats.items():
        if k in SERPER_FEATS:
            dfb.loc[dfb.index[idx], k] = v

def enrich_with_serper_pool(dfb: pd.DataFrame, gen: int, agent_id:
int, pool="train", name_col="name"):
    """Budget-maximizing enrichment: use df batch names first; if not
enough, pull from global pool."""
    for c in SERPER_FEATS:
        if c not in dfb.columns: dfb[c] = 0
    if not ENABLE_SERPER or name_col not in dfb.columns:
```

```
        return dfb, 0
    remaining = _BUDGETS[pool].total - _BUDGETS[pool].used_total
    target_per_agent = SERPER_PER_AGENT_PER_GEN if pool=="train" else
min(50, remaining)
    target_per_agent = max(FORCE_MIN_ENRICH_PER_AGENT if pool=="train"
else 0, target_per_agent)
    cand = (
        dfb[[name_col]].assign(_idx=np.arange(len(dfb)))
        .dropna()
    )
    cand = cand[cand[name_col].astype(str).str.strip() != ""]
    if cand.empty: cand = pd.DataFrame({name_col:[], "_idx":[]})
    cand = cand.drop_duplicates(subset=[name_col])
    cand = cand.sample(frac=1.0, random_state=agent_id + 991) if
len(cand)>0 else cand
    filled = 0
    def _try_name(comp, idx=None):
        nonlocal filled
        kwargs = {
            "budget_name": pool,
            "gen": gen if pool=="train" else None,
            "agent_id": agent_id if pool=="train" else None,
            "force_min": (pool=="train" and filled <
FORCE_MIN_ENRICH_PER_AGENT)
        }
        data = serper_news(comp, **kwargs)
        if data.get("_spent"):
            if pool == "train":
                _seen_train.add(comp);
_enriched_train_names.add(comp.lower())
            else:
                _seen_test.add(comp)
        feats = extract_serper_features(comp, budget_name=None)
        if idx is not None:
            _apply_feats_rowwise(dfb, idx, feats)
        filled += 1
    for _, row in cand.iterrows():
        if filled >= target_per_agent: break
        if _BUDGETS[pool].used_total >= _BUDGETS[pool].total: break
        comp = str(row[name_col]).strip()
        if comp == "": continue
        if _cache_has(comp):
            continue
        _try_name(comp, int(row["_idx"]))
    if filled < target_per_agent and _BUDGETS[pool].used_total <
_BUDGETS[pool].total:
        need = min(target_per_agent - filled, _BUDGETS[pool].total -
_BUDGETS[pool].used_total)
        global_pool = _global_train_names if pool=="train" else
```

```
_global_test_names
        exclude = _seen_train if pool=="train" else _seen_test
        fresh = _next_fresh_names(global_pool, need, exclude=exclude)
        for comp in fresh:
            if filled >= target_per_agent: break
            if _BUDGETS[pool].used_total >= _BUDGETS[pool].total:
break
            _try_name(comp, None)
    return dfb, filled

print("\n=== Real-Time PED with Serper (TRAIN, target ~600) ===")
try:
    _agents_rt = agents
except NameError:
    _agents_rt = [PEDAgent(i, random_dna()) for i in range(20)]


=== Real-Time PED with Serper (TRAIN, target ~600) ===

for gen in range(N_GENERATIONS_RT):
    print(f"[RT/TRAIN] Gen {gen+1}/{N_GENERATIONS_RT} | Train budget
used: {_BUDGETS['train'].used_total}/{_BUDGETS['train'].total}")
    for i, agent in enumerate(_agents_rt):
        if 'GA_REAL_RATIO' in globals() and np.random.rand() <
GA_REAL_RATIO and len(train_master) > 0:
            dfb = train_master.sample(AGENT_BATCH_RT, replace=True,
random_state=gen*111 + i).copy()
        else:
            dfb = synthetic.sample(AGENT_BATCH_RT, replace=True,
random_state=gen*222 + i).copy()
        if 'name' in dfb.columns:
            dfb = dfb[dfb['name'].astype(str).str.strip() !=
""].copy()
        dfb, filled_n = enrich_with_serper_pool(dfb, gen,
getattr(agent, 'id', i), pool="train")
        Xb = build_X_with_serper(dfb)
        yb = dfb[target].values
        sectors = dfb['category_code'].values if 'category_code' in
dfb else np.full(len(dfb), np.nan)
        agent.train(Xb, yb)

[RT/TRAIN] Gen 1/3 | Train budget used: 0/580
[RT/TRAIN] Gen 2/3 | Train budget used: 232/580
[RT/TRAIN] Gen 3/3 | Train budget used: 464/580

print(f"[RT/TRAIN DONE] Train budget used:
{_BUDGETS['train'].used_total}/{_BUDGETS['train'].total}")

[RT/TRAIN DONE] Train budget used: 580/580
```

```
def _make_holdout(master_df: pd.DataFrame, n=9000):
    df = master_df.copy()
    if 'name' in df.columns:
        df = df[df['name'].astype(str).str.strip() != ""]
        df = df[~df['name'].str.lower().isin({x.lower() for x in
_enriched_train_names})]
    if 'exit_flag' in df.columns:
        df = df[df['exit_flag'].notnull()]
    return df.sample(min(n, len(df)), random_state=2025).copy()

def _bulk_enrich_holdout(hold: pd.DataFrame):
    if 'name' not in hold.columns or hold.empty: return hold
    names = hold['name'].astype(str).str.strip()
    idx_map = names.reset_index().groupby('name')
['index'].apply(list).to_dict()
    for comp, idxs in idx_map.items():
        if _BUDGETS["test"].used_total >= _BUDGETS["test"].total:
break
        if comp == "" or _cache_has(comp): continue
        _ = serper_news(comp, budget_name="test")
        feats = extract_serper_features(comp)
        for idx in idxs:
            for k,v in feats.items():
                if k in SERPER_FEATS:
                    hold.loc[idx, k] = v
    if _BUDGETS["test"].used_total < _BUDGETS["test"].total:
        need = _BUDGETS["test"].total - _BUDGETS["test"].used_total
        fresh = _next_fresh_names(_global_test_names, need, exclude={n
for n in names if n})
        extra_rows = []
        for comp in fresh:
            if _BUDGETS["test"].used_total >= _BUDGETS["test"].total:
break
            _ = serper_news(comp, budget_name="test")
            feats = extract_serper_features(comp)
            row = {c: 0 for c in SERPER_FEATS}
            row.update({k:v for k,v in feats.items() if k in
SERPER_FEATS})
            row['name'] = comp
            row['category_code'] =
hold['category_code'].mode().iloc[0] if 'category_code' in
hold.columns and not hold['category_code'].dropna().empty else 0
            row['exit_flag'] = 0 if 'exit_flag' not in hold.columns
else hold['exit_flag'].mode().iloc[0]
            extra_rows.append(row)
        if extra_rows:
            hold = pd.concat([hold, pd.DataFrame(extra_rows)],
ignore_index=True)
    for c in SERPER_FEATS:
        if c not in hold.columns: hold[c] = 0
```

```
        hold[c] = pd.to_numeric(hold[c], errors='coerce').fillna(0)
    return hold

def evaluate_agents_with_serper_holdout(master_df, agents_list,
n=9000):
    hold = _make_holdout(master_df, n=n)
    hold = _bulk_enrich_holdout(hold)
    X_hold = build_X_with_serper(hold)
    y_hold = hold[target].astype(int).values if 'exit_flag' in
hold.columns else np.zeros(len(hold), dtype=int)
    results = []
    for agent in agents_list:
        agent.train(build_X_with_serper(train_master),
train_master[target].values)
        preds, probs = agent.predict(X_hold,
hold['category_code'].values if 'category_code' in hold else
np.full(len(hold), np.nan))
        acc = accuracy_score(y_hold, preds) if len(set(y_hold))>1 else
np.nan
        auc = None
        try:
            auc = roc_auc_score(y_hold, probs)
        except Exception:
            pass
        results.append({"agent_id": getattr(agent, 'id', -1),
"accuracy": acc, "roc_auc": auc})
    res_df = pd.DataFrame(results).sort_values(["accuracy","roc_auc"],
ascending=[False, False])
    print("\n=== EVAL (Hold-out + Serper) ===")
    print(res_df.head(10))
    print(f"[RT/TEST DONE] Test budget used:
{_BUDGETS['test'].used_total}/{_BUDGETS['test'].total}")
    return res_df, hold

eval_results, eval_hold =
evaluate_agents_with_serper_holdout(train_master, _agents_rt, n=9000)
```

```
=== EVAL (Hold-out + Serper) ===
    agent_id  accuracy   roc_auc
16        30  1.000000  1.000000
19        33  1.000000  1.000000
0          6  1.000000  1.000000
2          5  1.000000  1.000000
7         21  1.000000  1.000000
11        25  1.000000  1.000000
13        27  1.000000  1.000000
6         20  0.999778  0.999999
3          2  0.999667  1.000000
```

```
8          22  0.997444  0.999529
[RT/TEST DONE] Test budget used: 1200/1200
```

```
train_accs = []
test_accs  = []

for gen in range(N_GENERATIONS_RT):
    print(f"[RT/TRAIN] Gen {gen+1}/{N_GENERATIONS_RT} | Train budget
used: {_BUDGETS['train'].used_total}/{_BUDGETS['train'].total}")
    for i, agent in enumerate(_agents_rt):
        dfb = train_master.sample(AGENT_BATCH_RT, replace=True,
random_state=gen*111 + i).copy()
        dfb = dfb[dfb['name'].astype(str).str.strip() != ""]
        dfb, _ = enrich_with_serper_pool(dfb, gen, getattr(agent,
'id', i), pool="train")
        Xb = build_X_with_serper(dfb)
        yb = dfb[target].values
        sectors = dfb['category_code'].values if 'category_code' in
dfb else np.full(len(dfb), np.nan)
        agent.train(Xb, yb)
        preds, _ = agent.predict(Xb, sectors)
        acc = accuracy_score(yb, preds)
        train_accs.append(acc)
        print(f"[TRAIN] Agent {getattr(agent,'id',i):02d}
acc={acc:.3f}")
```

```
[RT/TRAIN] Gen 1/3 | Train budget used: 580/580
[TRAIN] Agent 06 acc=1.000
[TRAIN] Agent 17 acc=1.000
[TRAIN] Agent 05 acc=1.000
[TRAIN] Agent 02 acc=1.000
[TRAIN] Agent 20 acc=1.000
[TRAIN] Agent 20 acc=0.999
[TRAIN] Agent 20 acc=1.000
[TRAIN] Agent 21 acc=1.000
[TRAIN] Agent 22 acc=1.000
[TRAIN] Agent 23 acc=0.998
[TRAIN] Agent 24 acc=0.999
[TRAIN] Agent 25 acc=1.000
[TRAIN] Agent 26 acc=1.000
[TRAIN] Agent 27 acc=1.000
[TRAIN] Agent 28 acc=0.999
[TRAIN] Agent 29 acc=0.999
[TRAIN] Agent 30 acc=1.000
[TRAIN] Agent 31 acc=1.000
[TRAIN] Agent 32 acc=1.000
[TRAIN] Agent 33 acc=1.000
[RT/TRAIN] Gen 2/3 | Train budget used: 580/580
[TRAIN] Agent 06 acc=1.000
[TRAIN] Agent 17 acc=0.999
```

```
[TRAIN] Agent 05 acc=1.000
[TRAIN] Agent 02 acc=1.000
[TRAIN] Agent 20 acc=1.000
[TRAIN] Agent 20 acc=1.000
[TRAIN] Agent 20 acc=1.000
[TRAIN] Agent 21 acc=1.000
[TRAIN] Agent 22 acc=1.000
[TRAIN] Agent 23 acc=0.997
[TRAIN] Agent 24 acc=1.000
[TRAIN] Agent 25 acc=1.000
[TRAIN] Agent 26 acc=1.000
[TRAIN] Agent 27 acc=1.000
[TRAIN] Agent 28 acc=0.999
[TRAIN] Agent 29 acc=1.000
[TRAIN] Agent 30 acc=1.000
[TRAIN] Agent 31 acc=0.999
[TRAIN] Agent 32 acc=1.000
[TRAIN] Agent 33 acc=1.000
[RT/TRAIN] Gen 3/3 | Train budget used: 580/580
[TRAIN] Agent 06 acc=1.000
[TRAIN] Agent 17 acc=0.999
[TRAIN] Agent 05 acc=1.000
[TRAIN] Agent 02 acc=1.000
[TRAIN] Agent 20 acc=1.000
[TRAIN] Agent 20 acc=1.000
[TRAIN] Agent 20 acc=1.000
[TRAIN] Agent 21 acc=1.000
[TRAIN] Agent 22 acc=0.999
[TRAIN] Agent 23 acc=0.998
[TRAIN] Agent 24 acc=1.000
[TRAIN] Agent 25 acc=1.000
[TRAIN] Agent 26 acc=1.000
[TRAIN] Agent 27 acc=1.000
[TRAIN] Agent 28 acc=1.000
[TRAIN] Agent 29 acc=0.999
[TRAIN] Agent 30 acc=1.000
[TRAIN] Agent 31 acc=1.000
[TRAIN] Agent 32 acc=0.999
[TRAIN] Agent 33 acc=1.000
```

```
print(f"[RT/TRAIN DONE] Train budget used:
{_BUDGETS['train'].used_total}/{_BUDGETS['train'].total}")
print(f"Total TRAIN accuracy: {np.mean(train_accs):.4f}")
```

```
[RT/TRAIN DONE] Train budget used: 580/580
Total TRAIN accuracy: 0.9995
```

```
for gen in range(N_GENERATIONS_RT):
    print(f"[RT/TEST] Gen {gen+1}/{N_GENERATIONS_RT} | Test budget
used: {_BUDGETS['test'].used_total}/{_BUDGETS['test'].total}")
```

```
    for i, agent in enumerate(_agents_rt):
        dfb = synthetic.sample(AGENT_BATCH_RT, replace=True,
random_state=gen*222 + i).copy()
        dfb = dfb[dfb['name'].astype(str).str.strip() != ""]
        dfb, _ = enrich_with_serper_pool(dfb, gen, getattr(agent,
'id', i), pool="test")
        Xb = build_X_with_serper(dfb)
        yb = dfb[target].values
        sectors = dfb['category_code'].values if 'category_code' in
dfb else np.full(len(dfb), np.nan)
        preds, _ = agent.predict(Xb, sectors)
        acc = accuracy_score(yb, preds)
        test_accs.append(acc)
        print(f"[TEST] Agent {getattr(agent,'id',i):02d}
acc={acc:.3f}")
```

```
[RT/TEST] Gen 1/3 | Test budget used: 1200/1200
[TEST] Agent 06 acc=0.631
[TEST] Agent 17 acc=0.803
[TEST] Agent 05 acc=0.794
[TEST] Agent 02 acc=0.800
[TEST] Agent 20 acc=0.797
[TEST] Agent 20 acc=0.823
[TEST] Agent 20 acc=0.801
[TEST] Agent 21 acc=0.818
[TEST] Agent 22 acc=0.806
[TEST] Agent 23 acc=0.549
[TEST] Agent 24 acc=0.804
[TEST] Agent 25 acc=0.759
[TEST] Agent 26 acc=0.820
[TEST] Agent 27 acc=0.796
[TEST] Agent 28 acc=0.778
[TEST] Agent 29 acc=0.700
[TEST] Agent 30 acc=0.789
[TEST] Agent 31 acc=0.783
[TEST] Agent 32 acc=0.693
[TEST] Agent 33 acc=0.810
[RT/TEST] Gen 2/3 | Test budget used: 1200/1200
[TEST] Agent 06 acc=0.620
[TEST] Agent 17 acc=0.795
[TEST] Agent 05 acc=0.791
[TEST] Agent 02 acc=0.806
[TEST] Agent 20 acc=0.794
[TEST] Agent 20 acc=0.803
[TEST] Agent 20 acc=0.789
[TEST] Agent 21 acc=0.817
[TEST] Agent 22 acc=0.808
[TEST] Agent 23 acc=0.527
[TEST] Agent 24 acc=0.807
[TEST] Agent 25 acc=0.748
```

```
[TEST] Agent 26 acc=0.812
[TEST] Agent 27 acc=0.806
[TEST] Agent 28 acc=0.778
[TEST] Agent 29 acc=0.701
[TEST] Agent 30 acc=0.782
[TEST] Agent 31 acc=0.775
[TEST] Agent 32 acc=0.698
[TEST] Agent 33 acc=0.811
[RT/TEST] Gen 3/3 | Test budget used: 1200/1200
[TEST] Agent 06 acc=0.618
[TEST] Agent 17 acc=0.798
[TEST] Agent 05 acc=0.797
[TEST] Agent 02 acc=0.797
[TEST] Agent 20 acc=0.784
[TEST] Agent 20 acc=0.797
[TEST] Agent 20 acc=0.803
[TEST] Agent 21 acc=0.823
[TEST] Agent 22 acc=0.805
[TEST] Agent 23 acc=0.539
[TEST] Agent 24 acc=0.815
[TEST] Agent 25 acc=0.746
[TEST] Agent 26 acc=0.805
[TEST] Agent 27 acc=0.809
[TEST] Agent 28 acc=0.792
[TEST] Agent 29 acc=0.690
[TEST] Agent 30 acc=0.785
[TEST] Agent 31 acc=0.789
[TEST] Agent 32 acc=0.710
[TEST] Agent 33 acc=0.808
```

```
print(f"[RT/TEST DONE] Test budget used:
{_BUDGETS['test'].used_total}/{_BUDGETS['test'].total}")
print(f"Total TEST accuracy: {np.mean(test_accs):.4f}")
```

```
[RT/TEST DONE] Test budget used: 1200/1200
Total TEST accuracy: 0.7656
```

```
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score, roc_auc_score,
precision_recall_fscore_support
from sklearn.utils import resample

_required = ["train_master", "_agents_rt", "build_X_with_serper",
"SERPER_FEATS", "target"]
for _k in _required:
    if _k not in globals():
        raise ValueError(f"Missing global `{_k}`. Run the earlier
cells that define the PED agents + Serper features first.")
```

```
if "_make_holdout" not in globals():
    def _make_holdout(master_df: pd.DataFrame, n=9000):
        df = master_df.copy()
        if 'name' in df.columns:
            df = df[df['name'].astype(str).str.strip() != ""]
        if 'exit_flag' in df.columns:
            df = df[df['exit_flag'].notnull()]
        return df.sample(min(n, len(df)), random_state=2025).copy()

MIN_CONF_ERROR = 0.65
UPSAMPLE_FACTOR = 5
VAL_SPLIT = 0.15
RANDOM_STATE = 2025

_NONNEG_COLS = [

'company_age','num_offices','funding_total_usd','total_raised_usd','nu
m_rounds',

'funding_per_year','avg_round_size_usd','investors_count','num_founder
s','num_employees',

'num_current_employees','milestones_count','news_count','recent_news_c
ount','vc_deal_count',
    'num_funds','funds_total_usd'
]

def _clip_nonneg(df, cols):
    for c in cols:
        if c in df.columns:
            df[c] = pd.to_numeric(df[c], errors='coerce').fillna(0)
            df[c] = df[c].clip(lower=0)
    return df

def _biased_probs(agent, X, sectors):
    """Match agent.predict() logic but return post-bias
probabilities."""
    probs = agent.model.predict_proba(X)[:, 1]
    if getattr(agent, "sector_col", None) in train_master.columns and
getattr(agent, "sector_bias", None):
        bias_vec = []
        for s in sectors:
            if pd.isna(s):
                bias_vec.append(1.0)
            else:
                try:
                    bias_vec.append(agent.sector_bias.get(int(s),
1.0))
                except Exception:
                    bias_vec.append(1.0)
```

```
        bias_vec = np.array(bias_vec, dtype=float)
        probs = np.clip(probs * bias_vec, 0, 1)
    return probs

def _tune_threshold(agent, df):
    """Grid-search threshold on a small validation slice using
*biased* probs (no data leakage)."""
    if df.empty:
        return agent.threshold
    df = df.sample(frac=1.0,
random_state=RANDOM_STATE).reset_index(drop=True)
    n_val = max(1000, int(len(df) * VAL_SPLIT))
    val = df.iloc[:n_val].copy()
    Xv = build_X_with_serper(val)
    yv = val[target].astype(int).values
    sectors = val['category_code'].values if 'category_code' in val
else np.full(len(val), np.nan)
    probs = _biased_probs(agent, Xv, sectors)
    grid = np.linspace(0.2, 0.8, 25)
    best_f1, best_thr = -1.0, float(getattr(agent, "threshold", 0.5))
    for thr in grid:
        preds = (probs >= thr).astype(int)
        _, _, f1, _ = precision_recall_fscore_support(yv, preds,
average='binary', zero_division=0.0)
        if f1 > best_f1:
            best_f1, best_thr = f1, float(thr)
    return best_thr

def _collect_failures(hold_df, agents_list):
    """Run all agents on holdout and collect confident failures (no
leakage back into this same holdout later)."""
    logs = []
    X_hold = build_X_with_serper(hold_df)
    sectors = hold_df['category_code'].values if 'category_code' in
hold_df else np.full(len(hold_df), np.nan)
    y_true = hold_df[target].astype(int).values
    for ag in agents_list:
        ag.train(build_X_with_serper(train_master),
train_master[target].values)
        preds, probs = ag.predict(X_hold, sectors)
        wrong = preds != y_true
        conf  = np.where(preds == 1, probs, 1.0 - probs)
        sel   = wrong & (conf >= MIN_CONF_ERROR)
        if sel.any():
            df_err = hold_df.loc[sel].copy()
            df_err['agent_id']   = getattr(ag, 'id', -1)
            df_err['pred_label'] = preds[sel]
            df_err['pred_prob']  = probs[sel]
            df_err['conf']       = conf[sel]
            logs.append(df_err)
```

```
    return pd.concat(logs, ignore_index=True) if logs else
pd.DataFrame()

def _upsample_failures(fails_df):
    """Build an augmented training set emphasizing failures, with non-
neg clipping and SERPER feat hygiene."""
    if fails_df.empty:
        return train_master.copy()
    if 'name' in fails_df.columns:
        _cap = 5
        fails_df = (fails_df
                    .sort_values('conf', ascending=False)
                    .groupby('name', group_keys=False)
                    .head(_cap)
                    .reset_index(drop=True))
    fails_df = _clip_nonneg(fails_df.copy(), _NONNEG_COLS)
    for c in SERPER_FEATS:
        if c not in fails_df.columns:
            fails_df[c] = 0
        fails_df[c] = pd.to_numeric(fails_df[c],
errors='coerce').fillna(0)
    ups = resample(
        fails_df,
        replace=True,
        n_samples=max(len(fails_df) * UPSAMPLE_FACTOR, len(fails_df)),
        random_state=RANDOM_STATE
    )
    base = train_master.sample(
        min(len(train_master), max(5000, len(ups)//2)),
        random_state=RANDOM_STATE
    ).copy()
    mix = pd.concat([base, ups], ignore_index=True)
    mix = _clip_nonneg(mix, _NONNEG_COLS)
    for c in SERPER_FEATS:
        mix[c] = pd.to_numeric(mix.get(c, 0),
errors='coerce').fillna(0)
    mix = mix.sample(frac=1.0,
random_state=RANDOM_STATE).reset_index(drop=True)
    return mix

hold_A = _make_holdout(train_master, n=9000)
hold_B = _make_holdout(train_master, n=9000)

failures = _collect_failures(hold_A, _agents_rt)
print(f"[FAILURES] Collected {len(failures)} high-confidence mistakes
from Holdout-A.")
```

```
[FAILURES] Collected 207 high-confidence mistakes from Holdout-A.
```

```
aug_train = _upsample_failures(failures)
```

```
X_aug   = build_X_with_serper(aug_train)
y_aug   = aug_train[target].astype(int).values
X_holdB = build_X_with_serper(hold_B)
y_holdB = hold_B[target].astype(int).values
sectors_B = hold_B['category_code'].values if 'category_code' in
hold_B else np.full(len(hold_B), np.nan)

post_rows = []
for ag in _agents_rt:
    ag.train(X_aug, y_aug)
    ag.threshold = _tune_threshold(ag, aug_train)
    predsB, probsB = ag.predict(X_holdB, sectors_B)
    accB = accuracy_score(y_holdB, predsB) if len(set(y_holdB)) > 1
else np.nan
    try:
        aucB = roc_auc_score(y_holdB, probsB)
    except Exception:
        aucB = np.nan
    post_rows.append({
        "agent_id": getattr(ag, 'id', -1),
        "acc_post": accB,
        "auc_post": aucB,
        "thr": ag.threshold
    })

post_df = pd.DataFrame(post_rows).sort_values(["acc_post",
"auc_post"], ascending=[False, False])
print("\n=== AFTER FAILURE RETRAIN (Evaluated on Holdout-B) ===")
print(post_df.head(10))
print(f"Mean acc post: {np.nanmean(post_df['acc_post']):.4f} | Mean
AUC post: {np.nanmean(post_df['auc_post']):.4f}")


=== AFTER FAILURE RETRAIN (Evaluated on Holdout-B) ===
    agent_id  acc_post  auc_post    thr
19        33  0.999556  0.999967  0.225
7         21  0.999333  0.999970  0.200
11        25  0.999333  0.999941  0.200
4         20  0.999222  0.999332  0.450
0          6  0.999111  0.999982  0.200
13        27  0.999000  0.999970  0.200
2          5  0.999000  0.999950  0.275
3          2  0.998778  0.999939  0.225
6         20  0.998667  0.999926  0.200
5         20  0.998667  0.999764  0.475
Mean acc post: 0.9987 | Mean AUC post: 0.9995

if '_BUDGETS' in globals() and isinstance(_BUDGETS, dict):
    train_used = _BUDGETS.get('train').used_total if
_BUDGETS.get('train') else None
```

```
    train_tot  = _BUDGETS.get('train').total      if
_BUDGETS.get('train') else None
    test_used  = _BUDGETS.get('test').used_total  if
_BUDGETS.get('test') else None
    test_tot   = _BUDGETS.get('test').total       if
_BUDGETS.get('test') else None
    print(f"[Budgets] Train used: {train_used}/{train_tot} | Test
used: {test_used}/{test_tot}")
```

```
[Budgets] Train used: 580/580 | Test used: 1200/1200
```