assignment1-ql2510

February 22, 2024

1 Your Uni : ql2510. (Also change Uni in the title of your notebook)

2 Your AI Model Share Username: liqiankun

3 Your Full name : Qiankun Li

4 Link to your Public Github repository with Final report : https://github.com/artemlevinh/Predicting-Happiness-Based-on-Economical-Educational-and-Geographical-Data

5 World Happiness Classification Competition

Goals : - Understand how the models function - Understand what the parameters control - Share your models to a centralized leaderboard - Learn from the model experimentation process - Make a good looking notebook report - Upload as a personal project on Github

Overall Steps: 1. Get data in and set up X_train / X_test / y_train 2. Preprocess data using Sklearn Column Transformer/ Write and Save Preprocessor function 3. Fit model on preprocessed data and save preprocessor function and model 4. Generate predictions from X_test data and submit model to competition 5. Repeat submission process to improve place on leaderboard

[4]:
```
#install aimodelshare library
! pip install aimodelshare --upgrade
```

```
Collecting aimodelshare
  Downloading aimodelshare-0.1.11-py3-none-any.whl (975 kB)
                           975.8/975.8

kB 6.5 MB/s eta 0:00:00
Collecting boto3==1.26.69 (from aimodelshare)
  Downloading boto3-1.26.69-py3-none-any.whl (132 kB)
                           132.7/132.7

kB 7.3 MB/s eta 0:00:00
Collecting botocore==1.29.82 (from aimodelshare)
  Downloading botocore-1.29.82-py3-none-any.whl (10.5 MB)
                           10.5/10.5 MB
```

```
23.4 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn==1.2.2 in
/usr/local/lib/python3.10/dist-packages (from aimodelshare) (1.2.2)
Collecting onnx==1.13.1 (from aimodelshare)
  Downloading
onnx-1.13.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13.5 MB)
                            13.5/13.5 MB
36.4 MB/s eta 0:00:00
Collecting onnxconverter-common>=1.7.0 (from aimodelshare)
  Downloading onnxconverter_common-1.14.0-py2.py3-none-any.whl (84 kB)
                            84.5/84.5 kB
8.8 MB/s eta 0:00:00
Requirement already satisfied: regex in /usr/local/lib/python3.10/dist-
packages (from aimodelshare) (2023.12.25)
Collecting keras2onnx>=1.7.0 (from aimodelshare)
  Downloading keras2onnx-1.7.0-py3-none-any.whl (96 kB)
                            96.3/96.3 kB
11.1 MB/s eta 0:00:00
Requirement already satisfied: tensorflow>=2.12 in
/usr/local/lib/python3.10/dist-packages (from aimodelshare) (2.15.0)
Collecting tf2onnx (from aimodelshare)
  Downloading tf2onnx-1.16.1-py3-none-any.whl (455 kB)
                            455.8/455.8

kB 41.7 MB/s eta 0:00:00
Collecting skl2onnx>=1.14.0 (from aimodelshare)
  Downloading skl2onnx-1.16.0-py2.py3-none-any.whl (298 kB)
                            298.5/298.5

kB 26.8 MB/s eta 0:00:00
Collecting onnxruntime>=1.7.0 (from aimodelshare)
  Downloading
onnxruntime-1.17.0-cp310-cp310-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl
(6.8 MB)
                            6.8/6.8 MB
72.0 MB/s eta 0:00:00
Requirement already satisfied: torch>=1.8.1 in
/usr/local/lib/python3.10/dist-packages (from aimodelshare) (2.1.0+cu121)
Collecting pydot==1.3.0 (from aimodelshare)
  Downloading pydot-1.3.0-py2.py3-none-any.whl (18 kB)
Collecting importlib-resources==5.10.0 (from aimodelshare)
  Downloading importlib_resources-5.10.0-py3-none-any.whl (34 kB)
Collecting onnxmltools>=1.6.1 (from aimodelshare)
  Downloading onnxmltools-1.12.0-py2.py3-none-any.whl (329 kB)
                            329.0/329.0

kB 31.4 MB/s eta 0:00:00
Collecting Pympler==0.9 (from aimodelshare)
  Downloading Pympler-0.9.tar.gz (178 kB)
```

```
                           178.4/178.4

kB 20.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Collecting docker==5.0.0 (from aimodelshare)
  Downloading docker-5.0.0-py2.py3-none-any.whl (146 kB)
                           147.0/147.0

kB 16.9 MB/s eta 0:00:00
Collecting wget==3.2 (from aimodelshare)
  Downloading wget-3.2.zip (10 kB)
  Preparing metadata (setup.py) … done
Collecting PyJWT>=2.4.0 (from aimodelshare)
  Downloading PyJWT-2.8.0-py3-none-any.whl (22 kB)
Requirement already satisfied: seaborn>=0.11.2 in
/usr/local/lib/python3.10/dist-packages (from aimodelshare) (0.13.1)
Requirement already satisfied: astunparse==1.6.3 in
/usr/local/lib/python3.10/dist-packages (from aimodelshare) (1.6.3)
Collecting shortuuid>=1.0.8 (from aimodelshare)
  Downloading shortuuid-1.0.11-py3-none-any.whl (10 kB)
Requirement already satisfied: psutil>=5.9.1 in /usr/local/lib/python3.10/dist-
packages (from aimodelshare) (5.9.5)
Requirement already satisfied: pathlib>=1.0.1 in /usr/local/lib/python3.10/dist-
packages (from aimodelshare) (1.0.1)
Requirement already satisfied: protobuf>=3.20.1 in
/usr/local/lib/python3.10/dist-packages (from aimodelshare) (3.20.3)
Collecting dill (from aimodelshare)
  Downloading dill-0.3.8-py3-none-any.whl (116 kB)
                           116.3/116.3

kB 12.8 MB/s eta 0:00:00
Collecting scikeras (from aimodelshare)
  Downloading scikeras-0.12.0-py3-none-any.whl (27 kB)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse==1.6.3->aimodelshare)
(0.42.0)
Requirement already satisfied: six<2.0,>=1.6.1 in
/usr/local/lib/python3.10/dist-packages (from astunparse==1.6.3->aimodelshare)
(1.16.0)
Collecting jmespath<2.0.0,>=0.7.1 (from boto3==1.26.69->aimodelshare)
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting s3transfer<0.7.0,>=0.6.0 (from boto3==1.26.69->aimodelshare)
  Downloading s3transfer-0.6.2-py3-none-any.whl (79 kB)
                           79.8/79.8 kB
5.5 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in
/usr/local/lib/python3.10/dist-packages (from botocore==1.29.82->aimodelshare)
(2.8.2)
Collecting urllib3<1.27,>=1.25.4 (from botocore==1.29.82->aimodelshare)
```

```
  Downloading urllib3-1.26.18-py2.py3-none-any.whl (143 kB)
                           143.8/143.8

kB 16.3 MB/s eta 0:00:00
Requirement already satisfied: websocket-client>=0.32.0 in
/usr/local/lib/python3.10/dist-packages (from docker==5.0.0->aimodelshare)
(1.7.0)
Requirement already satisfied: requests!=2.18.0,>=2.14.2 in
/usr/local/lib/python3.10/dist-packages (from docker==5.0.0->aimodelshare)
(2.31.0)
Requirement already satisfied: numpy>=1.16.6 in /usr/local/lib/python3.10/dist-
packages (from onnx==1.13.1->aimodelshare) (1.25.2)
Requirement already satisfied: typing-extensions>=3.6.2.1 in
/usr/local/lib/python3.10/dist-packages (from onnx==1.13.1->aimodelshare)
(4.9.0)
Requirement already satisfied: pyparsing>=2.1.4 in
/usr/local/lib/python3.10/dist-packages (from pydot==1.3.0->aimodelshare)
(3.1.1)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn==1.2.2->aimodelshare) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn==1.2.2->aimodelshare) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn==1.2.2->aimodelshare)
(3.3.0)
Collecting fire (from keras2onnx>=1.7.0->aimodelshare)
  Downloading fire-0.5.0.tar.gz (88 kB)
                           88.3/88.3 kB
10.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from onnxconverter-common>=1.7.0->aimodelshare) (23.2)
Collecting protobuf>=3.20.1 (from aimodelshare)
  Downloading
protobuf-3.20.2-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.1
MB)
                           1.1/1.1 MB
5.2 MB/s eta 0:00:00
Collecting coloredlogs (from onnxruntime>=1.7.0->aimodelshare)
  Downloading coloredlogs-15.0.1-py2.py3-none-any.whl (46 kB)
                           46.0/46.0 kB
931.7 kB/s eta 0:00:00
Requirement already satisfied: flatbuffers in
/usr/local/lib/python3.10/dist-packages (from onnxruntime>=1.7.0->aimodelshare)
(23.5.26)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from onnxruntime>=1.7.0->aimodelshare) (1.12)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-
```

```
packages (from seaborn>=0.11.2->aimodelshare) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
/usr/local/lib/python3.10/dist-packages (from seaborn>=0.11.2->aimodelshare)
(3.7.1)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow>=2.12->aimodelshare) (1.4.0)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow>=2.12->aimodelshare) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(16.0.6)
Requirement already satisfied: ml-dtypes~=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(0.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(3.3.0)
INFO: pip is looking at multiple versions of tensorflow to determine which
version is compatible with other requirements. This could take a while.
Collecting tensorflow>=2.12 (from aimodelshare)
  Downloading tensorflow-2.15.0.post1-cp310-cp310-manylinux_2_17_x86_64.manylinu
x2014_x86_64.whl (475.2 MB)
                           475.2/475.2

MB 2.2 MB/s eta 0:00:00
  Downloading
tensorflow-2.14.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(489.9 MB)
                           489.9/489.9

MB 1.9 MB/s eta 0:00:00
  Downloading
tensorflow-2.14.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(489.8 MB)
                           489.8/489.8

MB 1.1 MB/s eta 0:00:00
  Downloading
tensorflow-2.13.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(479.7 MB)
                           479.7/479.7

MB 1.3 MB/s eta 0:00:00
```

```
Collecting gast<=0.4.0,>=0.2.1 (from tensorflow>=2.12->aimodelshare)
  Downloading gast-0.4.0-py3-none-any.whl (9.8 kB)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(1.60.1)
Collecting keras<2.14,>=2.13.1 (from tensorflow>=2.12->aimodelshare)
  Downloading keras-2.13.1-py3-none-any.whl (1.7 MB)
                           1.7/1.7 MB
55.3 MB/s eta 0:00:00
Collecting numpy>=1.16.6 (from onnx==1.13.1->aimodelshare)
  Downloading
numpy-1.24.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3
MB)
                           17.3/17.3 MB
40.6 MB/s eta 0:00:00
Collecting tensorflow>=2.12 (from aimodelshare)
  Downloading
tensorflow-2.13.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(524.1 MB)
                           524.1/524.1

MB 1.7 MB/s eta 0:00:00
  Downloading
tensorflow-2.12.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(585.9 MB)
                           585.9/585.9

MB 2.5 MB/s eta 0:00:00
Requirement already satisfied: jax>=0.3.15 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(0.4.23)
Collecting keras<2.13,>=2.12.0 (from tensorflow>=2.12->aimodelshare)
  Downloading keras-2.12.0-py2.py3-none-any.whl (1.7 MB)
                           1.7/1.7 MB
61.8 MB/s eta 0:00:00
Collecting tensorflow>=2.12 (from aimodelshare)
  Downloading
tensorflow-2.12.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(585.9 MB)
                           585.9/585.9

MB 2.2 MB/s eta 0:00:00
Collecting numpy>=1.16.6 (from onnx==1.13.1->aimodelshare)
  Downloading
numpy-1.23.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1
MB)
                           17.1/17.1 MB
41.7 MB/s eta 0:00:00
INFO: pip is looking at multiple versions of tensorflow to determine which
```

```
version is compatible with other requirements. This could take a while.
Collecting skl2onnx>=1.14.0 (from aimodelshare)
  Downloading skl2onnx-1.15.0-py2.py3-none-any.whl (294 kB)
                           294.7/294.7

kB 20.6 MB/s eta 0:00:00
INFO: This is taking longer than usual. You might need to provide the
dependency resolver with stricter constraints to reduce runtime. See
https://pip.pypa.io/warnings/backtracking for guidance. If you want to abort
this run, press Ctrl + C.
Collecting onnxruntime>=1.7.0 (from aimodelshare)
  Downloading
onnxruntime-1.16.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(6.4 MB)
                           6.4/6.4 MB
64.2 MB/s eta 0:00:00
  Downloading
onnxruntime-1.16.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(6.4 MB)
                           6.4/6.4 MB
51.7 MB/s eta 0:00:00
  Downloading
onnxruntime-1.16.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(6.2 MB)
                           6.2/6.2 MB
55.9 MB/s eta 0:00:00
  Downloading
onnxruntime-1.16.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(6.2 MB)
                           6.2/6.2 MB
61.1 MB/s eta 0:00:00
  Downloading
onnxruntime-1.15.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(5.9 MB)
                           5.9/5.9 MB
63.1 MB/s eta 0:00:00
  Downloading
onnxruntime-1.15.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(5.9 MB)
                           5.9/5.9 MB
39.5 MB/s eta 0:00:00
  Downloading onnxruntime-1.14.1-cp310-cp310-manylinux_2_27_x86_64.whl
(5.0 MB)
                           5.0/5.0 MB
47.4 MB/s eta 0:00:00
  Downloading onnxruntime-1.14.0-cp310-cp310-manylinux_2_27_x86_64.whl
(5.0 MB)
                           5.0/5.0 MB
```

```
55.2 MB/s eta 0:00:00
  Downloading onnxruntime-1.13.1-cp310-cp310-manylinux_2_27_x86_64.whl
(4.5 MB)
                           4.5/4.5 MB
70.2 MB/s eta 0:00:00
  Downloading onnxruntime-1.12.1-cp310-cp310-manylinux_2_27_x86_64.whl
(4.9 MB)
                           4.9/4.9 MB
41.4 MB/s eta 0:00:00
  Downloading onnxruntime-1.12.0-cp310-cp310-manylinux_2_27_x86_64.whl
(4.9 MB)
                           4.9/4.9 MB
16.2 MB/s eta 0:00:00
Collecting onnxmltools>=1.6.1 (from aimodelshare)
  Downloading onnxmltools-1.11.2-py2.py3-none-any.whl (322 kB)
                           322.5/322.5

kB 25.6 MB/s eta 0:00:00
Collecting onnxconverter-common>=1.7.0 (from aimodelshare)
  Downloading onnxconverter_common-1.13.0-py2.py3-none-any.whl (83 kB)
                           83.8/83.8 kB
8.3 MB/s eta 0:00:00
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(67.7.2)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(2.4.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(0.36.0)
Requirement already satisfied: tensorboard<2.16,>=2.15 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow>=2.12->aimodelshare)
(2.15.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from torch>=1.8.1->aimodelshare) (3.13.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch>=1.8.1->aimodelshare) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
```

```
(from torch>=1.8.1->aimodelshare) (3.1.3)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch>=1.8.1->aimodelshare) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-
packages (from torch>=1.8.1->aimodelshare) (2.1.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
matplotlib!=3.6.1,>=3.4->seaborn>=0.11.2->aimodelshare) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib!=3.6.1,>=3.4->seaborn>=0.11.2->aimodelshare) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from
matplotlib!=3.6.1,>=3.4->seaborn>=0.11.2->aimodelshare) (4.49.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
matplotlib!=3.6.1,>=3.4->seaborn>=0.11.2->aimodelshare) (1.4.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib!=3.6.1,>=3.4->seaborn>=0.11.2->aimodelshare) (9.4.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.2->seaborn>=0.11.2->aimodelshare) (2023.4)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from
requests!=2.18.0,>=2.14.2->docker==5.0.0->aimodelshare) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests!=2.18.0,>=2.14.2->docker==5.0.0->aimodelshare) (3.6)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from
requests!=2.18.0,>=2.14.2->docker==5.0.0->aimodelshare) (2024.2.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow>=2.12->aimodelshare) (2.27.0)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow>=2.12->aimodelshare) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow>=2.12->aimodelshare) (3.5.2)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow>=2.12->aimodelshare) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow>=2.12->aimodelshare) (3.0.1)
Collecting humanfriendly>=9.1 (from
coloredlogs->onnxruntime>=1.7.0->aimodelshare)
  Downloading humanfriendly-10.0-py2.py3-none-any.whl (86 kB)
                           86.8/86.8 kB
9.0 MB/s eta 0:00:00
```

```
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from
jinja2->torch>=1.8.1->aimodelshare) (2.1.5)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-
packages (from sympy->onnxruntime>=1.7.0->aimodelshare) (1.3.0)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow>=2.12->aimodelshare) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow>=2.12->aimodelshare) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-
packages (from google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow>=2.12->aimodelshare) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth-
oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow>=2.12->aimodelshare)
(1.3.1)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow>=2.12->aimodelshare) (0.5.1)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-
auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow>=2.12->aimodelshare)
(3.2.2)
Building wheels for collected packages: Pympler, wget, fire
  Building wheel for Pympler (setup.py) … done
  Created wheel for Pympler: filename=Pympler-0.9-py3-none-any.whl size=164823
sha256=6bd83209c91c08014bb415e7f495b01bd3eed81caee6179069bfe7509bb42349
  Stored in directory: /root/.cache/pip/wheels/82/8b/87/a090abb9a44f310f0126252e
5e291d4757c8f0520cf66e1eff
  Building wheel for wget (setup.py) … done
  Created wheel for wget: filename=wget-3.2-py3-none-any.whl size=9655
sha256=1ce09e6e9b4921a2722e6e29bd1db9e42e91c37abd7d7800cb337f7f2cffb201
  Stored in directory: /root/.cache/pip/wheels/8b/f1/7f/5c94f0a7a505ca1c81cd1d92
08ae2064675d97582078e6c769
  Building wheel for fire (setup.py) … done
  Created wheel for fire: filename=fire-0.5.0-py2.py3-none-any.whl size=116934
sha256=b8328a366431abd00e052c785621a6a3c54566a0ae1a3274fcc34b80eaabda5b
  Stored in directory: /root/.cache/pip/wheels/90/d4/f7/9404e5db0116bd4d43e5666e
aa3e70ab53723e1e3ea40c9a95
Successfully built Pympler wget fire
Installing collected packages: wget, Pympler, urllib3, shortuuid, PyJWT, pydot,
onnx, jmespath, importlib-resources, humanfriendly, fire, dill, onnxmltools,
onnxconverter-common, coloredlogs, botocore, tf2onnx, skl2onnx, scikeras,
s3transfer, onnxruntime, keras2onnx, docker, boto3, aimodelshare
  Attempting uninstall: urllib3
    Found existing installation: urllib3 2.0.7
    Uninstalling urllib3-2.0.7:
      Successfully uninstalled urllib3-2.0.7
  Attempting uninstall: PyJWT
    Found existing installation: PyJWT 2.3.0
    Uninstalling PyJWT-2.3.0:
      Successfully uninstalled PyJWT-2.3.0
  Attempting uninstall: pydot
    Found existing installation: pydot 1.4.2
    Uninstalling pydot-1.4.2:
      Successfully uninstalled pydot-1.4.2
  Attempting uninstall: importlib-resources
    Found existing installation: importlib-resources 6.1.1
    Uninstalling importlib-resources-6.1.1:
      Successfully uninstalled importlib-resources-6.1.1
Successfully installed PyJWT-2.8.0 Pympler-0.9 aimodelshare-0.1.11 boto3-1.26.69
botocore-1.29.82 coloredlogs-15.0.1 dill-0.3.8 docker-5.0.0 fire-0.5.0
humanfriendly-10.0 importlib-resources-5.10.0 jmespath-1.0.1 keras2onnx-1.7.0
onnx-1.13.1 onnxconverter-common-1.13.0 onnxmltools-1.12.0 onnxruntime-1.17.0
pydot-1.3.0 s3transfer-0.6.2 scikeras-0.12.0 shortuuid-1.0.11 skl2onnx-1.16.0
tf2onnx-1.16.1 urllib3-1.26.18 wget-3.2
```

5.1 0. Get data in and set up X_train, X_test, y_train objects

Instructions: Upload the world_happiness_competition_data.zip and newcountryvars.csv files on Colab by pressing the Folder icon on left tab (Shows "Files" on hovering), and then clicking the left-most file upload button (Shows "Upload to Session Storage" Alt text on hovering)

[7]:
```python
# Get competition data from course folder and unzip
# importing the zipfile module
from zipfile import ZipFile

# loading the temp.zip and creating a zip object
with ZipFile("/world_happiness_competition_data.zip", 'r') as zObject:

    # Extracting all the members of the zip
    # into a specific location.
    zObject.extractall()
```

[8]:
```python
# Load data
import pandas as pd
X_train = pd.read_csv('world_happiness_competition_data/X_train.csv')
X_test = pd.read_csv('world_happiness_competition_data/X_test.csv')
#y_test = pd.read_csv('world_happiness_competition_data/y_test.csv')
y_train = pd.read_csv('world_happiness_competition_data/y_train.csv')
y_train_labels = y_train.idxmax(axis=1) ## Examine what this does and write in␣
 ↪next cell

X_train.head()
```

```
[8]:  Country or region  GDP per capita  Social support  Healthy life expectancy  \
    0              Peru           0.960           1.274                    0.854
    1         Nicaragua           0.694           1.325                    0.835
    2            Greece           1.181           1.156                    0.999
    3             Qatar           1.684           1.313                    0.871
    4        Uzbekistan           0.745           1.529                    0.756

       Freedom to make life choices  Generosity  Perceptions of corruption  \
    0                         0.455       0.083                      0.027
    1                         0.435       0.200                      0.127
    2                         0.067       0.000                      0.034
    3                         0.555       0.220                      0.167
    4                         0.631       0.322                      0.240

             name    region                       sub-region  Terrorist_attacks
    0        Peru  Americas  Latin America and the Caribbean          18.000000
    1   Nicaragua  Americas  Latin America and the Caribbean         125.611111
    2      Greece    Europe                  Southern Europe         112.000000
    3       Qatar      Asia                     Western Asia          57.333333
    4  Uzbekistan      Asia                     Central Asia         125.611111
```

```
[9]: print(y_train)
```

```
    Average  High  Low  Very High  Very Low
0         1     0    0          0         0
1         0     1    0          0         0
2         1     0    0          0         0
3         0     0    0          1         0
4         0     1    0          0         0
..      …     …    …         …         …
83        1     0    0          0         0
84        0     0    1          0         0
85        0     0    0          1         0
86        1     0    0          0         0
87        0     0    1          0         0

[88 rows x 5 columns]
```

Write in the next cell what the y_train_labels = y_train.idxmax(axis=1) line does. What is the difference between y_train_labels and y_train?

```
[10]: # Your answer: According to Pandas documentation, idxmas returns the row label␣
      ↪of the maximum value. The result is the happiness values (label) of each␣
      ↪country. Since only the assigned happiness value has 1 and 0 is elsewhere.␣
      ↪Only the assigned happiness value is placed as the labels.
```

According to Pandas documentation, idxmas returns the row label of the maximum value. The result is the happiness values (label) of each country. Since only the assigned happiness value has

1 and 0 is elsewhere. Only the assigned happiness value is placed as the labels.

5.2 Add new data

```
[12]: # Truncated and cleaned up region data to merge (Week 4 folder)
      countrydata=pd.read_csv("/newcountryvars.csv")

      countrydata.head()
```

```
[12]:   country_name  population  population_below_poverty_line       hdi  \
      0        India  1339180127                           21.9  0.623559
      1      Nigeria   190886311                           70.0  0.527105
      2       Mexico   129163276                           46.2  0.761683
      3     Pakistan   197015955                           29.5  0.550354
      4   Bangladesh   164669751                           31.5  0.578824

         life_expectancy  expected_years_of_schooling  mean_years_of_schooling  \
      0           68.322                    11.696590                 6.298834
      1           53.057                     9.970482                 6.000000
      2           76.972                    13.299090                 8.554985
      3           66.365                     8.106910                 5.089460
      4           71.985                    10.178706                 5.241577

                  gni
      0   5663.474799
      1   5442.901264
      2  16383.106680
      3   5031.173074
      4   3341.490722
```

```
[13]: # Merge in new data to X_train and X_test by taking "Country or region" from␣
       ↪first table and "country_name" from 2nd table.

      X_train = X_train.merge(countrydata, how='left', left_on='Country or region',␣
       ↪right_on='country_name')
      X_test = X_test.merge(countrydata, how='left', left_on='Country or region',␣
       ↪right_on='country_name')

      # Drop the redundant columns after merging
      X_train.drop(columns=['Country or region', 'country_name'], inplace=True)
      X_test.drop(columns=['Country or region', 'country_name'], inplace=True)
```

```
[ ]: X_train.head(1)
```

6 EDA

[16]:
```
print(X_train.dtypes)
```

```
GDP per capita                      float64
Social support                      float64
Healthy life expectancy             float64
Freedom to make life choices        float64
Generosity                          float64
Perceptions of corruption           float64
name                                 object
region                               object
sub-region                           object
Terrorist_attacks                   float64
population                          float64
population_below_poverty_line       float64
hdi                                 float64
life_expectancy                     float64
expected_years_of_schooling         float64
mean_years_of_schooling             float64
gni                                 float64
dtype: object
```

Describe what you see above?

[17]:
```
## Your answer: we see the data type for each label (column). Most are float64␣
↪numbers, some are strings (names).
```

Find out the number and percentage of missing values in the table per column

[18]:
```
missing_values_count = X_train.isnull().sum()

# Percentage of missing values per column
missing_values_percentage = (missing_values_count / len(X_train)) * 100

# Combine into a DataFrame
missing_data = pd.DataFrame({'Missing Values': missing_values_count,␣
↪'Percentage': missing_values_percentage})
missing_data.sort_values(by='Missing Values', ascending=False, inplace=True)

print(missing_data)
```

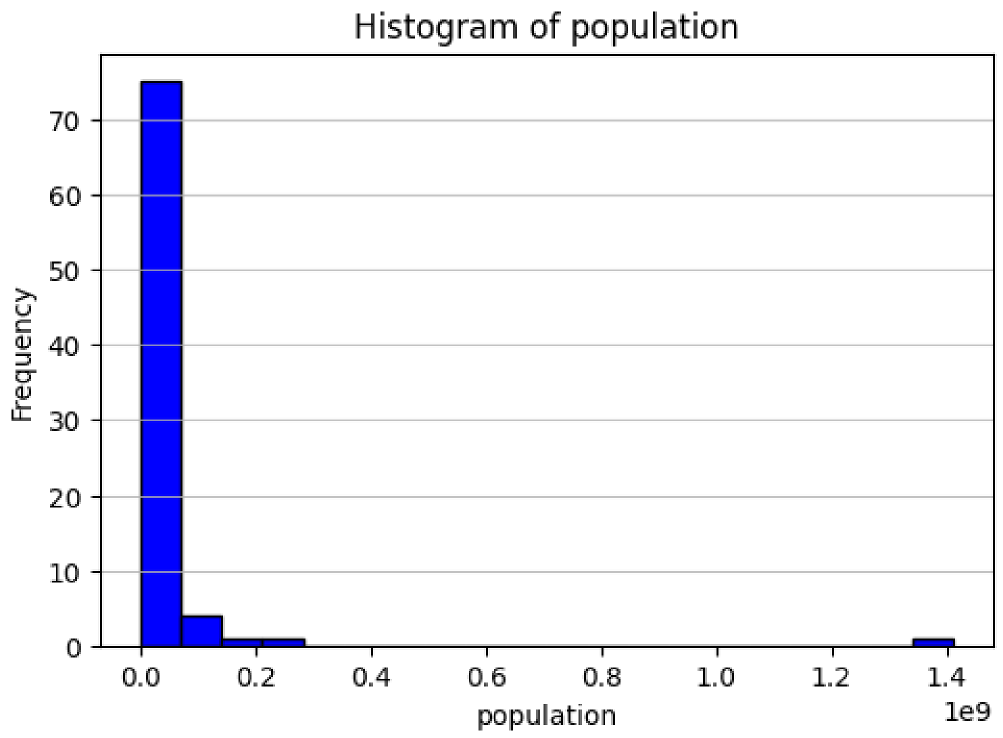
```
                               Missing Values  Percentage
population_below_poverty_line              15   17.045455
gni                                         6    6.818182
mean_years_of_schooling                     6    6.818182
expected_years_of_schooling                 6    6.818182
life_expectancy                             6    6.818182
hdi                                         6    6.818182
```

```
population                          6    6.818182
Terrorist_attacks                   0    0.000000
GDP per capita                      0    0.000000
Social support                      0    0.000000
region                              0    0.000000
name                                0    0.000000
Perceptions of corruption           0    0.000000
Generosity                          0    0.000000
Freedom to make life choices        0    0.000000
Healthy life expectancy             0    0.000000
sub-region                          0    0.000000
```

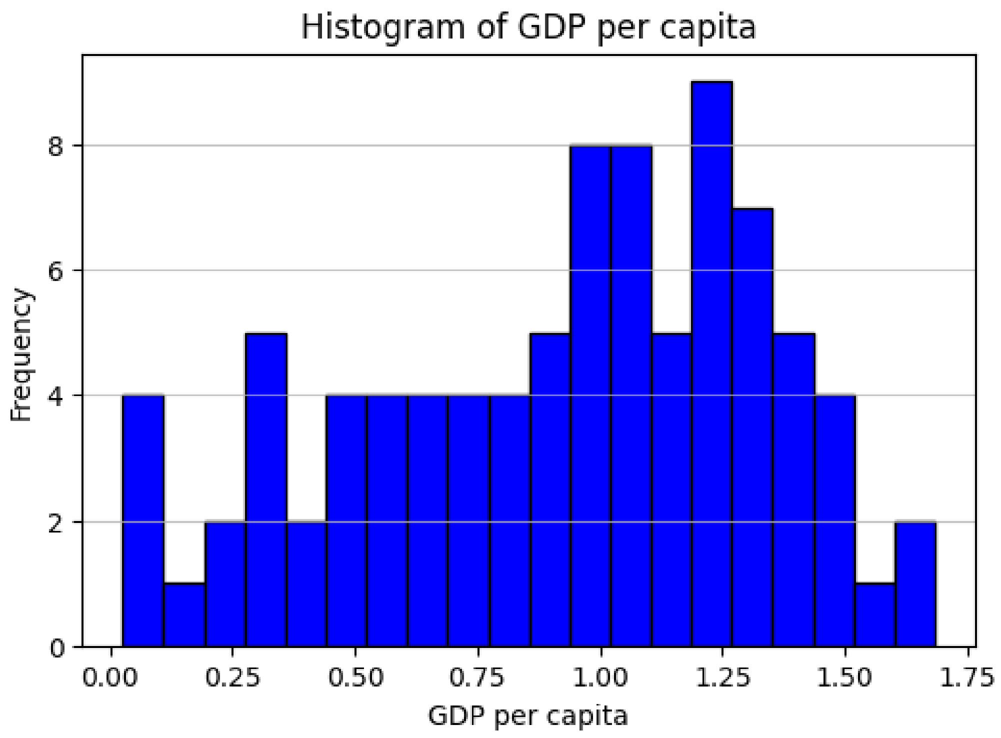
Plot the frequency distribution / histogram of some of the numerical features that you think are important

[19]:
```
#print(X_train)
```

[20]:
```
# Your plotting code here:
import matplotlib.pyplot as plt
feature = 'population'

# Plot histogram
plt.figure(figsize=(6, 4))
plt.hist(X_train[feature].dropna(), bins=20, color='blue', edgecolor='black')
plt.xlabel(feature)
plt.ylabel('Frequency')
plt.title('Histogram of ' + feature)
plt.grid(axis='y', alpha=0.75)
plt.show()
```

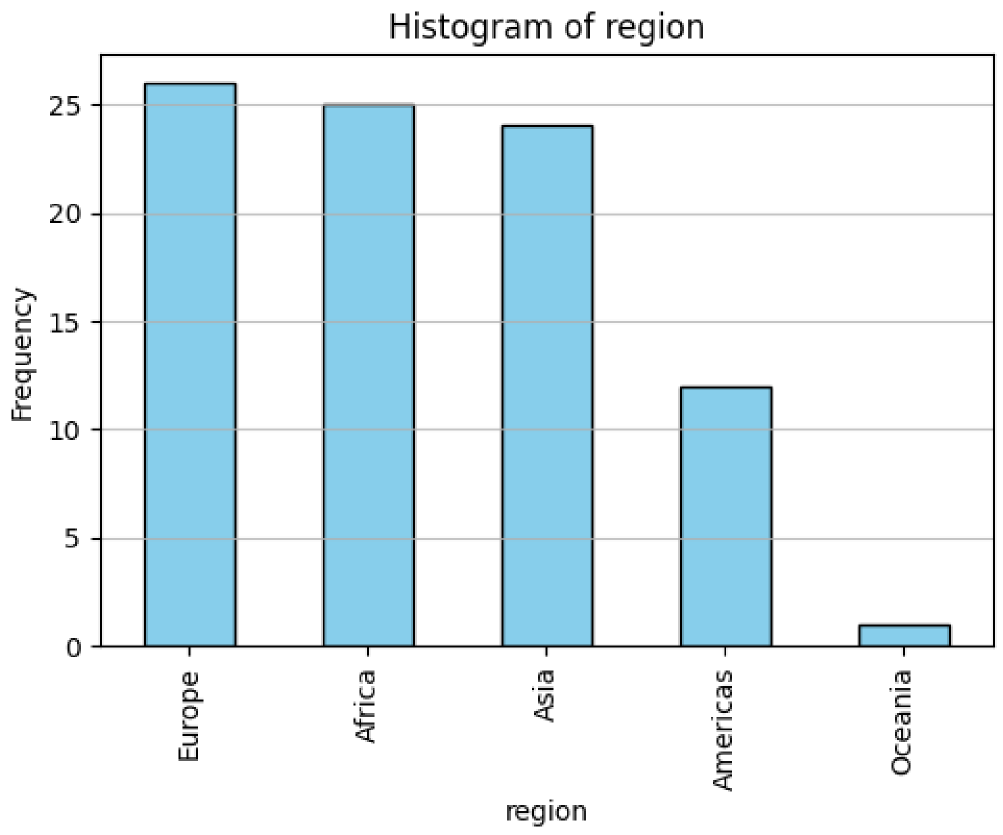


[21]:
```
feature = 'GDP per capita'

# Plot histogram
plt.figure(figsize=(6, 4))
plt.hist(X_train[feature].dropna(), bins=20, color='blue', edgecolor='black')
plt.xlabel(feature)
plt.ylabel('Frequency')
plt.title('Histogram of ' + feature)
plt.grid(axis='y', alpha=0.75)
plt.show()
```

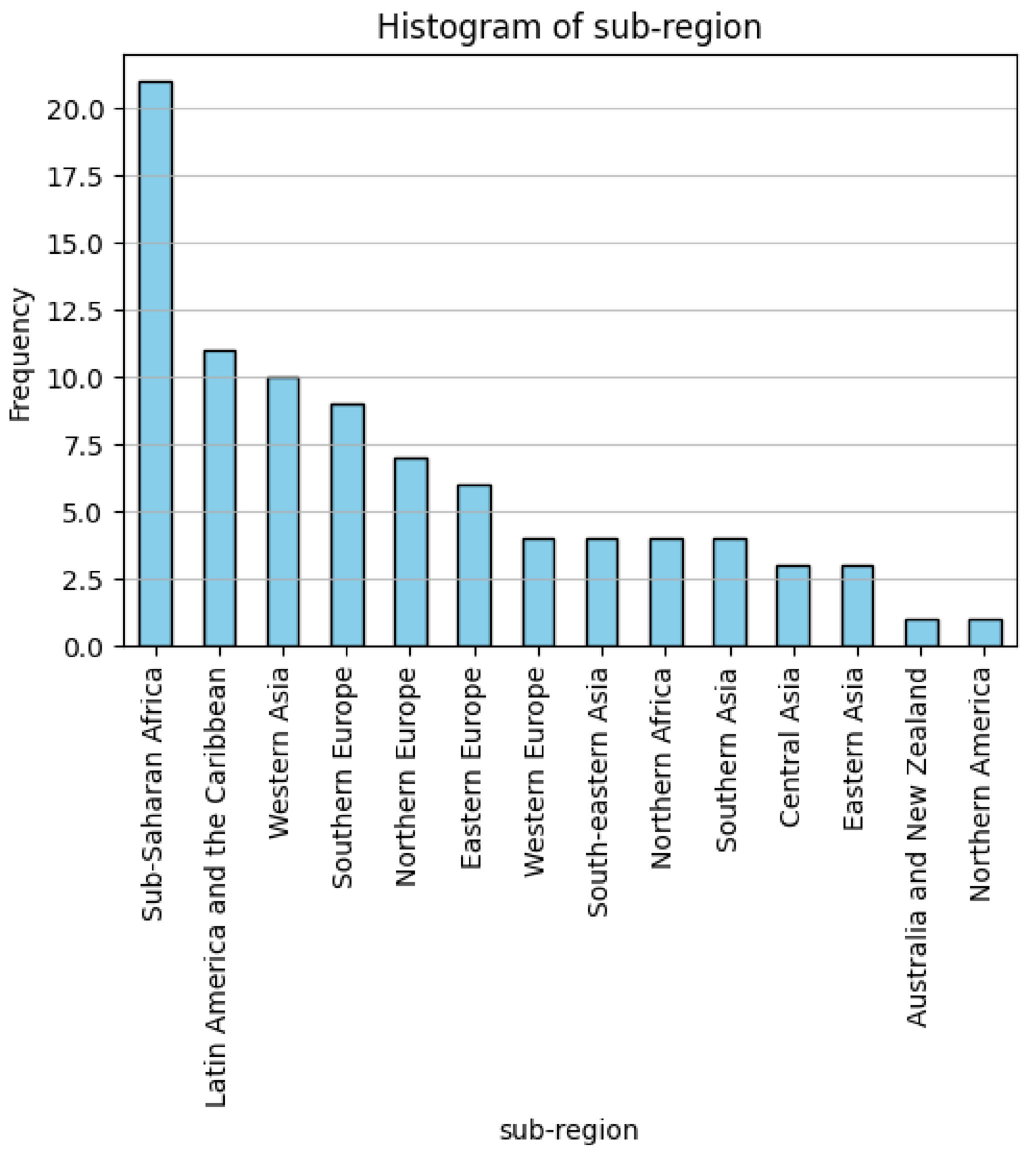


Plot the categorical variables and their distribution

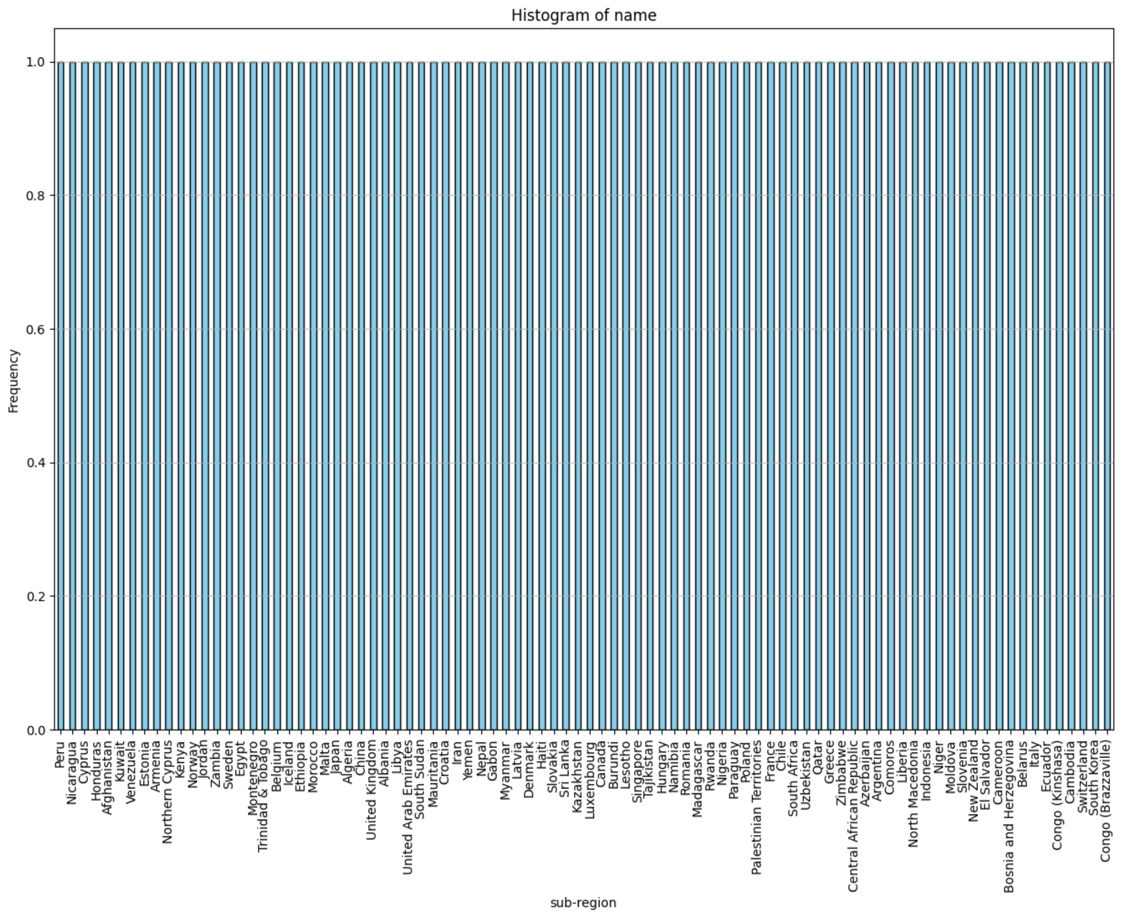
```
[22]: ##Your Plotting Code Here
      import matplotlib.pyplot as plt

      plt.figure(figsize=(6, 4))
      X_train['region'].value_counts().plot(kind='bar', color='skyblue',␣
       ↪edgecolor='black')
      plt.xlabel('region')
      plt.ylabel('Frequency')
      plt.title('Histogram of ' + 'region')
      plt.grid(axis='y', alpha=0.75)
      plt.show()
```



[23]:
```
plt.figure(figsize=(6, 4))
X_train['sub-region'].value_counts().plot(kind='bar', color='skyblue',␣
  ↪edgecolor='black')
plt.xlabel('sub-region')
plt.ylabel('Frequency')
plt.title('Histogram of ' + 'sub-region')
plt.grid(axis='y', alpha=0.75)
plt.show()
```



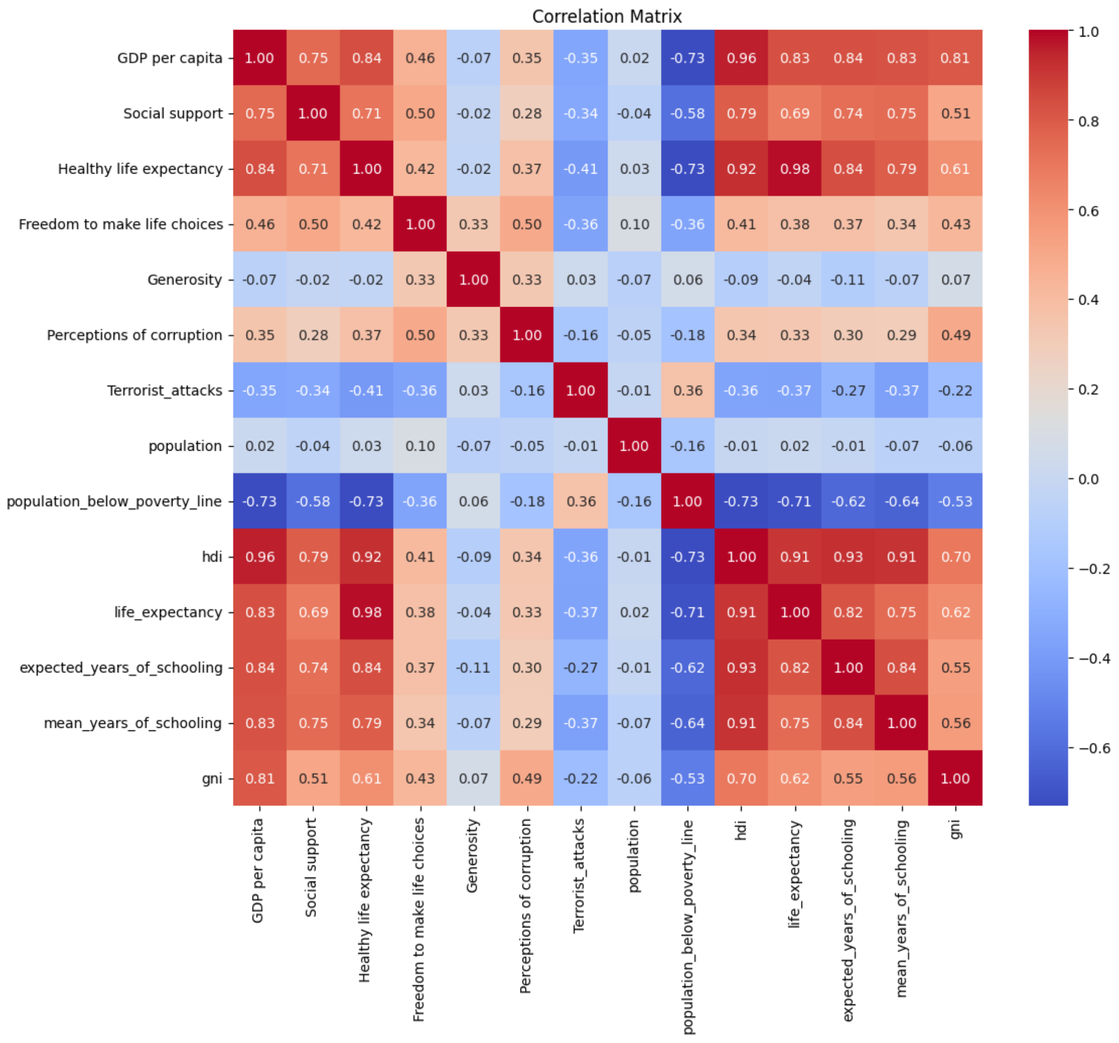
[24]:
```
plt.figure(figsize=(15, 10))
X_train['name'].value_counts().plot(kind='bar', color='skyblue',␣
 ↪edgecolor='black')
plt.xlabel('sub-region')
plt.ylabel('Frequency')
plt.title('Histogram of ' + 'name')
plt.grid(axis='y', alpha=0.75)
plt.show()
```

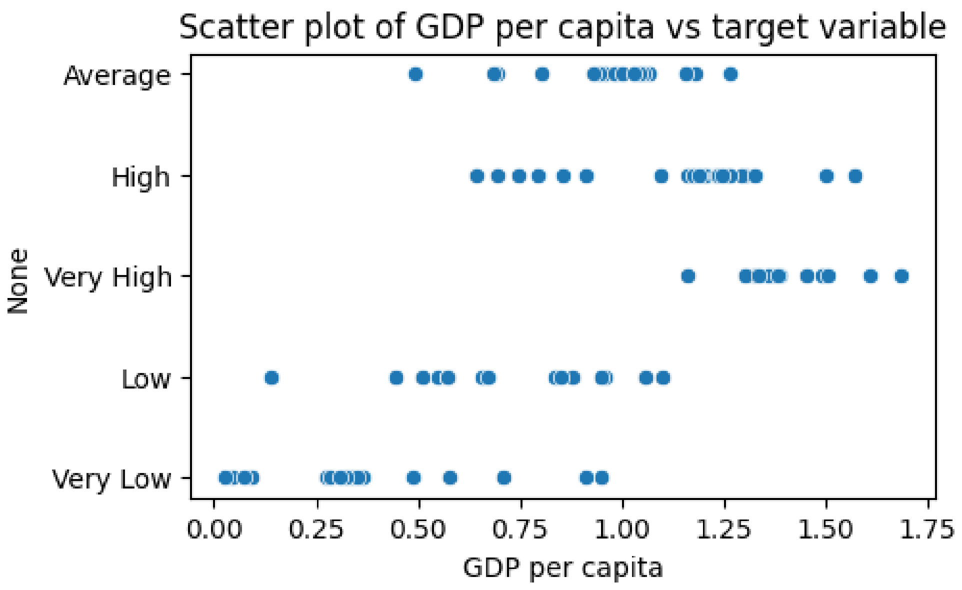


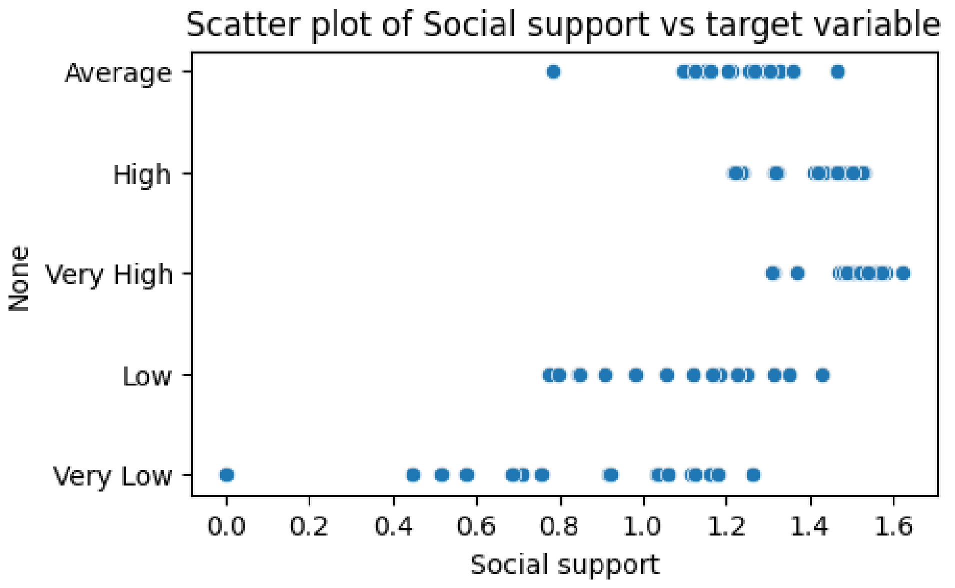
Explore relationships between variables (bivariate, etc), correlation tables, and how they associate with the target variable.

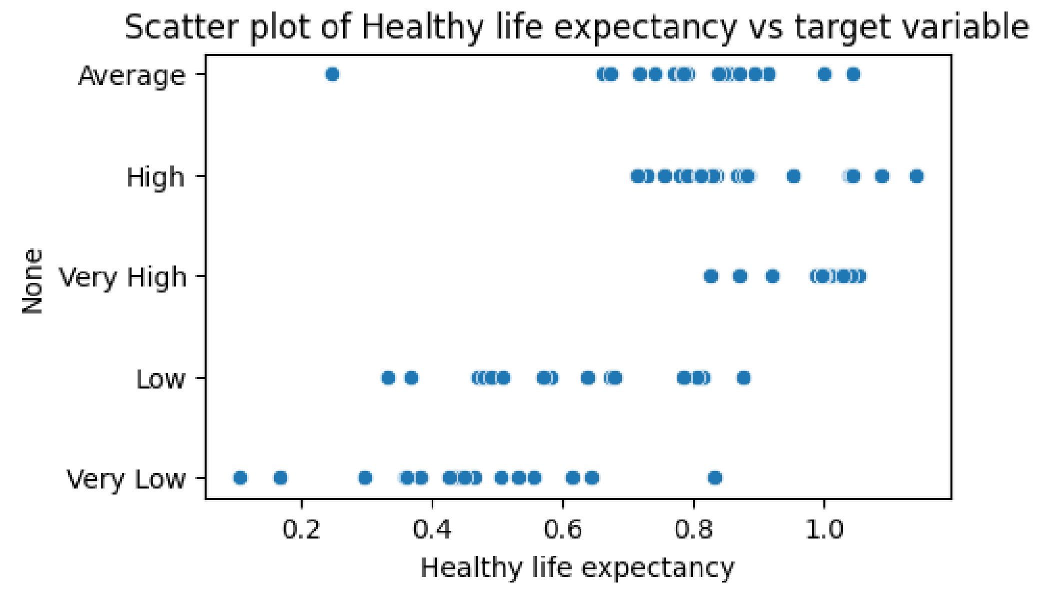
[25]:
```
# Your plotting code(s) here:
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the correlation matrix
correlation_matrix = X_train.corr()

# Plot the correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()

# Explore bivariate relationships of each value with the target variable
for column in X_train.columns:
    if X_train[column].dtype != 'object':
        plt.figure(figsize=(5, 3))
        sns.scatterplot(data=X_train, x=column, y=y_train_labels)
        plt.title('Scatter plot of ' + column + ' vs target variable')
        plt.show()
```

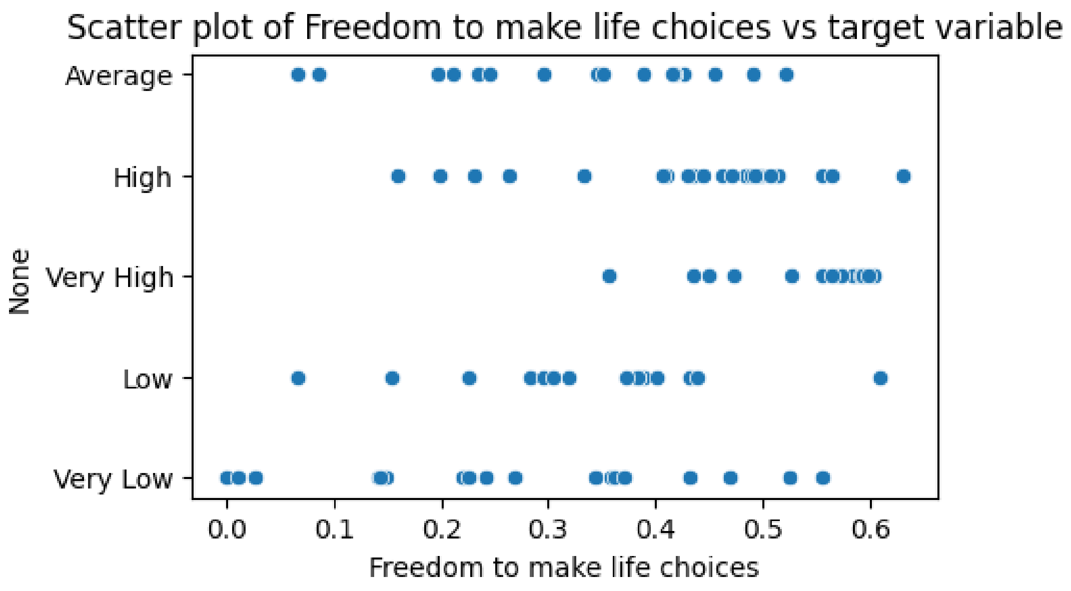
```
<ipython-input-25-cb26e25a1fe7>:6: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = X_train.corr()
```

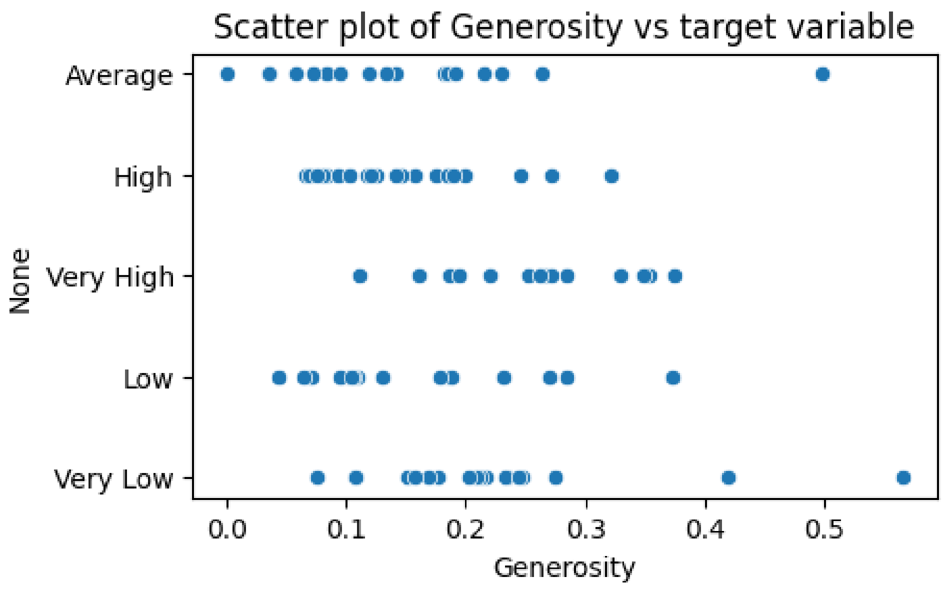


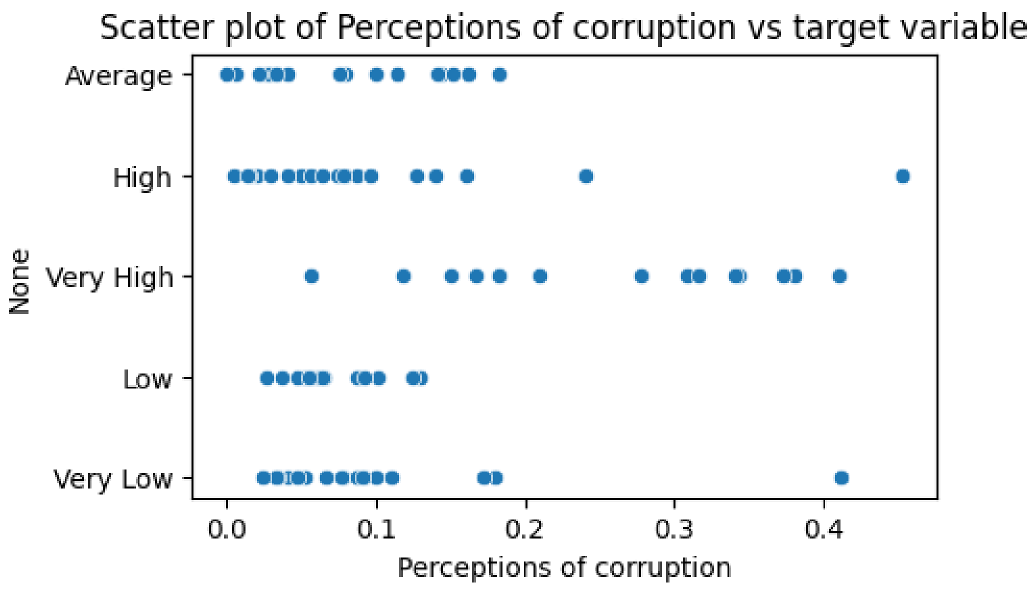

Scatter plot of GDP per capita vs target variable
Average
High
Very High
Low
Very Low
None
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
GDP per capita

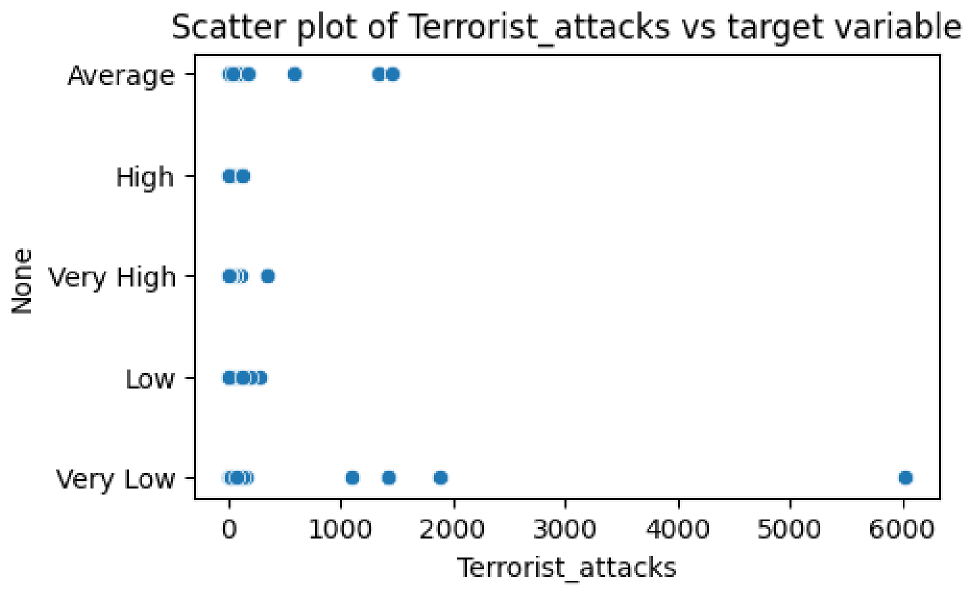

Scatter plot of Social support vs target variable
Average
High
Very High
Low
Very Low
None
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Social support

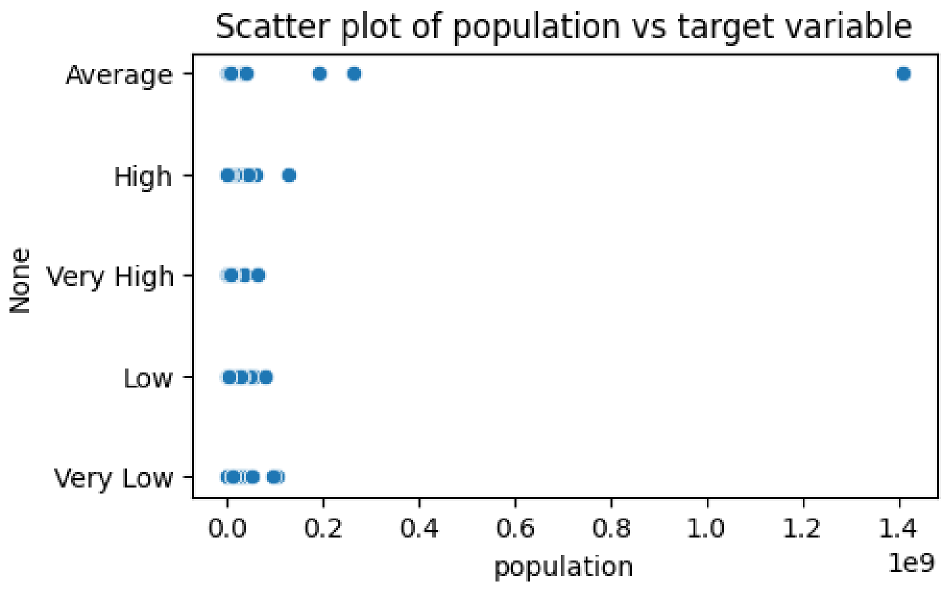

Scatter plot of Healthy life expectancy vs target variable
Average
High
Very High
Low
Very Low
None
0.2
0.4
0.6
0.8
1.0
Healthy life expectancy

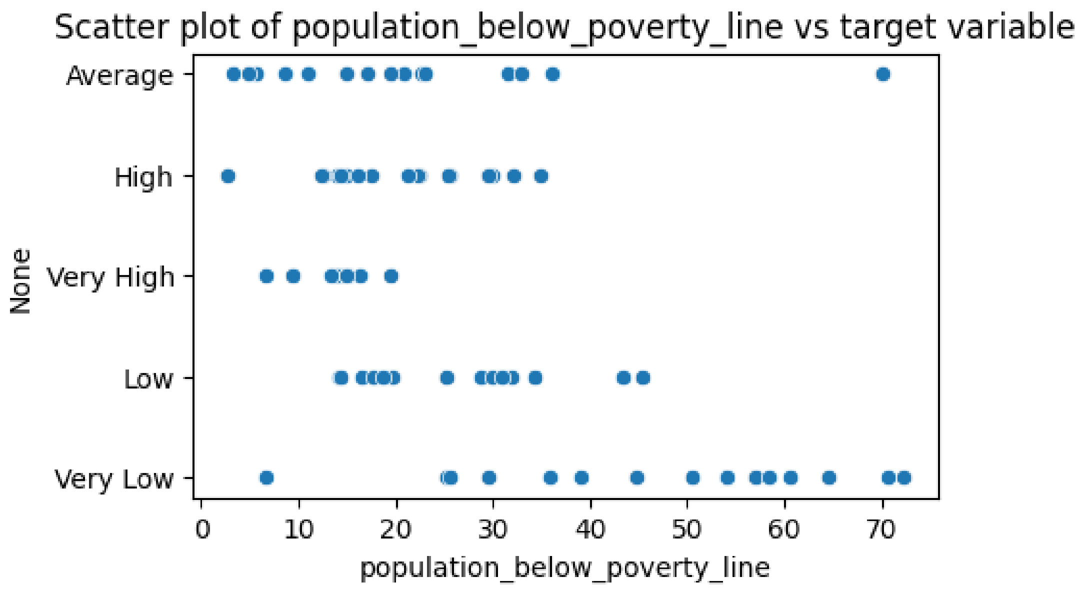

Scatter plot of Freedom to make life choices vs target variable
Average
High
Very High
Low
Very Low
None
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Freedom to make life choices

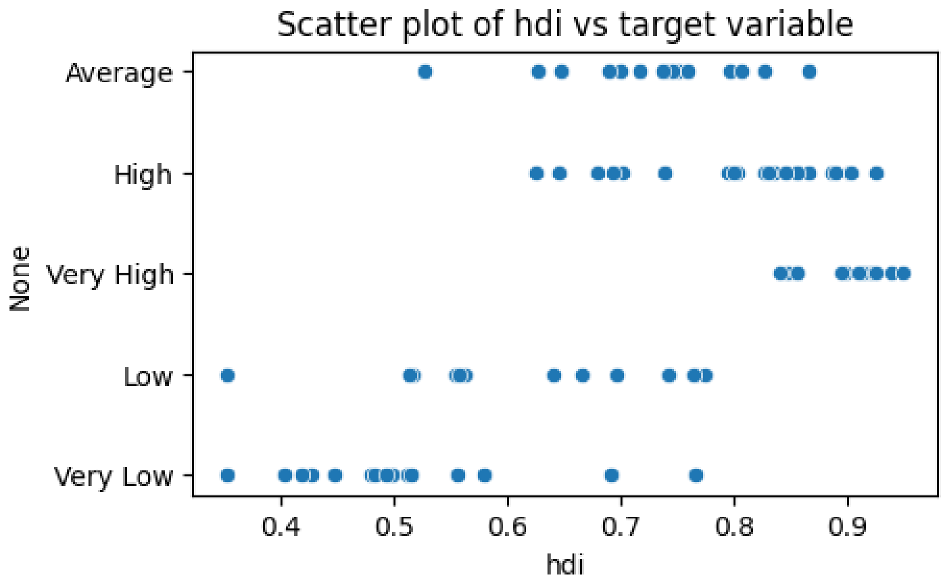

Scatter plot of Generosity vs target variable
Average
High
Very High
Low
Very Low
None
0.0
0.1
0.2
0.3
0.4
0.5
Generosity

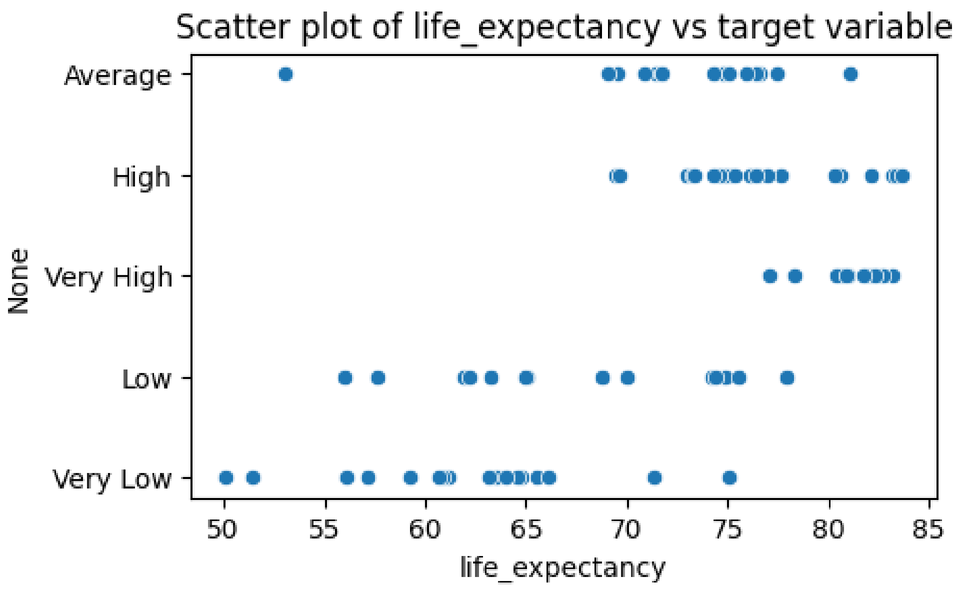

Scatter plot of Perceptions of corruption vs target variable
Average
High
Very High
Low
Very Low
None
0.0
0.1
0.2
0.3
0.4
Perceptions of corruption

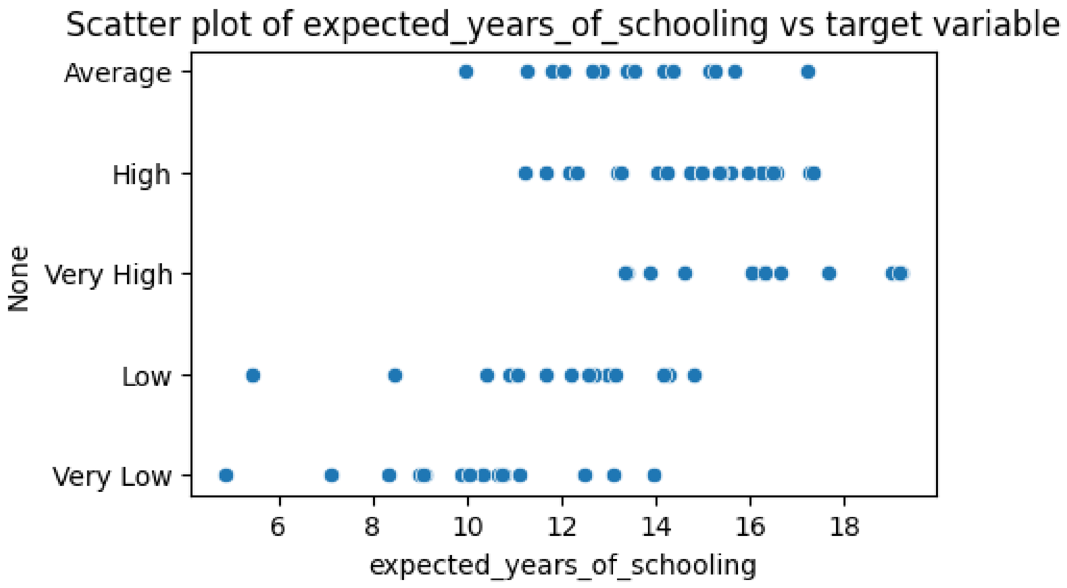

Scatter plot of Terrorist_attacks vs target variable
Average
High
Very High
Low
Very Low
None
0
1000
2000
3000
4000
5000
6000
Terrorist_attacks

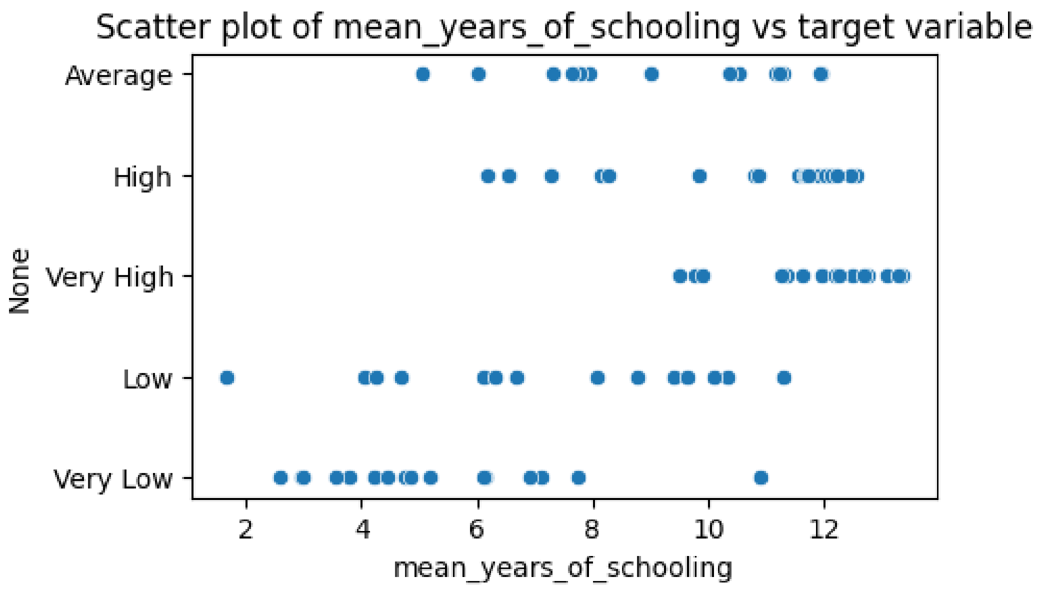

Scatter plot of population vs target variable
Average
High
Very High
Low
Very Low
None
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e9
population

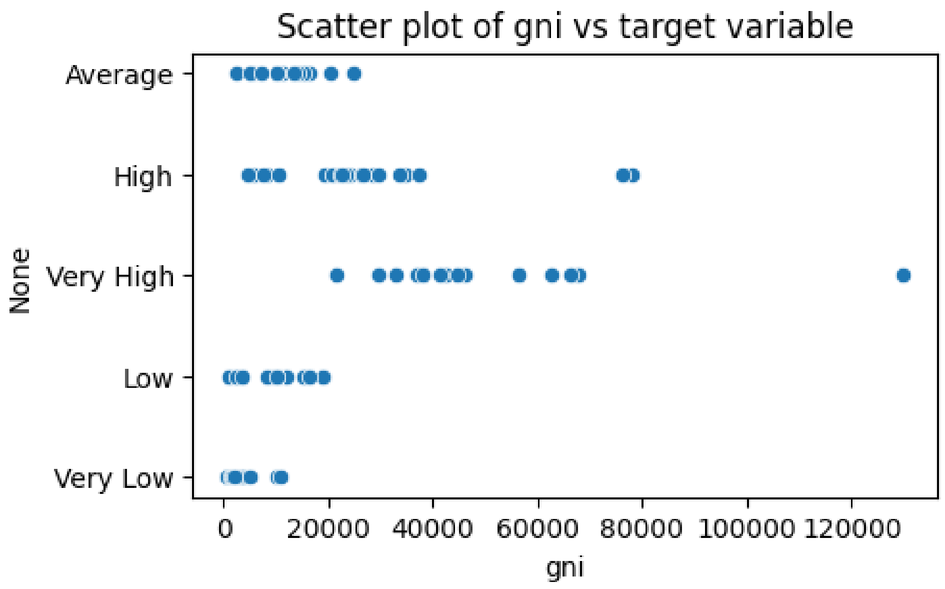

Scatter plot of population_below_poverty_line vs target variable
Average
High
Very High
Low
Very Low
None
0
10
20
30
40
50
60
70
population_below_poverty_line


Scatter plot of hdi vs target variable
Average
High
Very High
Low
Very Low
None
0.4
0.5
0.6
0.7
0.8
0.9
hdi


Scatter plot of life_expectancy vs target variable
Average
High
Very High
Low
Very Low
None
50
55
60
65
70
75
80
85
life_expectancy


Scatter plot of expected_years_of_schooling vs target variable
Average
High
Very High
Low
Very Low
None
6
8
10
12
14
16
18
expected_years_of_schooling





Write what you observed and your General comments on what should be done:

[26]:

```
# Your comments here: The bivariate relation between each variable and the␣
↪target vaiable is too obscure; there is no linear or other relation between␣
↪them. This is probably because there are many other variables that also␣
↪influence the target variable.Also we may observe that the covariance of␣
↪variables in the same aspect (for example health life and life expectancy)␣
↪are highly related, wherea data of irrelevant fields such as terrorist␣
↪attach and population, have low covariance.
```

The bivariate relation between each variable and the target vaiable is too obscure; there is no linear or other obvious relation between them. This is probably because there are many other variables that also influence the target variable. Also we may observe that the covariance of variables in the same aspect (for example health life and life expectancy) are highly related, wherea data of irrelevant fields such as terrorist attach and population, have low covariance.

##2. Preprocess data using Sklearn Column Transformer/ Write and Save Preprocessor function

[26]:
```

```

[27]:
```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer, make_column_transformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder

# Create the preprocessing pipelines for both numeric and categorical data.

numeric_features = X_train.select_dtypes(include=['float64', 'int64'])## Drop␣
↪all the non-numerical features from X_train
numeric_features=numeric_features.columns.tolist()

numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value=0)), ## Is this␣
↪good enough?
    ('scaler', StandardScaler())]) # You will need to describe why this is␣
↪being done in the next cell

categorical_features = ['region', 'sub-region']

#Replacing missing values with Modal value and then one hot encoding.
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),  # Replace missing␣
↪values with the most frequent value
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])  # One-hot encode the␣
↪categorical features, ignore unknown categories

# final preprocessor object set up with ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)])

#Fit your preprocessor object
preprocess=preprocessor.fit(X_train)
```

Describe step-by-step what we are doing above, and why? You are free to change how values are imputed. What change did you make if any, and why?

[28]:
```
## Your answer: this part transform our data into a np.array that we can "feed"␣
↪into our machine. For numerical values, we scale them such that the absolute␣
↪value of each scaled entry is roughly 1. There are two major concerns for␣
↪scaling: Faster Convergence: Scaling input features helps gradient descent␣
↪algorithms converge faster. When features are on different scales, the␣
↪optimization process may take longer to find the optimal weights. Numerical␣
↪Stability: Large differences in the scales of input features can lead to␣
↪numerical stability issues in the model. Scaling helps to avoid these issues␣
↪and ensures that the optimization algorithm behaves predictably. For␣
↪cetegorical features (strings), we convert them into onehot vectors. Note␣
↪that for missing values, I replace the N/A with the most frequent one.
```

This part transform our data into a np.array that we can "feed" into our machine. For numerical values, we scale them such that the absolute value of each scaled entry is roughly 1. There are two major concerns for scaling: Faster Convergence: Scaling input features helps gradient descent algorithms converge faster. When features are on different scales, the optimization process may take longer to find the optimal weights. Numerical Stability: Large differences in the scales of input features can lead to numerical stability issues in the model. Scaling helps to avoid these issues and ensures that the optimization algorithm behaves predictably. For cetegorical features (strings), we convert them into onehot vectors. Note that for missing values, I replace the N/A with the most frequent one.

[29]:
```
print(X_train.head(1))
```

```
   GDP per capita  Social support  Healthy life expectancy  \
0            0.96           1.274                    0.854

   Freedom to make life choices  Generosity  Perceptions of corruption  name  \
0                         0.455       0.083                      0.027  Peru

     region                       sub-region  Terrorist_attacks  population  \
0  Americas  Latin America and the Caribbean               18.0  32165485.0

   population_below_poverty_line       hdi  life_expectancy  \
0                           22.7  0.739749           74.814

   expected_years_of_schooling  mean_years_of_schooling          gni
0                     13.38634                  9.01347  11294.84033
```

[30]:
```python
# Write function to transform data with preprocessor

def preprocessor(data):
    data.drop(['region', 'name'], axis=1)
    preprocessed_data=preprocess.transform(data)
    return preprocessed_data
```

What are the differences between the "preprocessor" object, the "preprocess" object, the "preprocessor" function, and the "preprocessed_data" that is returned finally?

[31]:
```python
## Your Answer: preprocessor defines the properties (attributes), and␣
↪specifying the behaviors (methods) of data. Preprocess is a flow (work␣
↪structure) that we process data. The preprocessor function is the function␣
↪we call when we need to transform data. The proprocessed data is the numeric␣
↪value that the preprocess function returns. The data is scaled and converted.
```

[32]:
```python
# check shape of X data after preprocessing it using our new function
preprocessor(X_train).shape
```

[32]:
```
(88, 33)
```

[33]:
```python
preprocessor(X_train)
```

[33]:
```
array([[ 0.10444464,  0.20603101,  0.45831144, …,  0.        ,
         0.        ,  0.        ],
       [-0.54388211,  0.38059533,  0.37791099, …,  0.        ,
         0.        ,  0.        ],
       [ 0.64309205, -0.19786289,  1.07189386, …,  0.        ,
         0.        ,  0.        ],
       …,
       [ 1.01356448,  1.10965806,  1.05919905, …,  0.        ,
         0.        ,  0.        ],
       [ 0.27261963, -0.30397062,  0.62334396, …,  0.        ,
         0.        ,  0.        ],
       [-0.5950658 , -1.4198131 , -1.00582315, …,  1.        ,
         0.        ,  0.        ]])
```

##3. Fit model on preprocessed data and save preprocessor function and model

[34]:
```python
print(preprocessor(X_train))
```

```
[[ 0.10444464  0.20603101  0.45831144 …  0.          0.
   0.        ]
 [-0.54388211  0.38059533  0.37791099 …  0.          0.
   0.        ]
 [ 0.64309205 -0.19786289  1.07189386 …  0.          0.
   0.        ]
 …
 [ 1.01356448  1.10965806  1.05919905 …  0.          0.
   0.        ]
 [ 0.27261963 -0.30397062  0.62334396 …  0.          0.
   0.        ]
 [-0.5950658  -1.4198131  -1.00582315 …  1.          0.
   0.        ]]
```

[34]:

```

```

[35]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

X_train_processed = preprocessor(X_train)
# Define the Random Forest model
model_1 = RandomForestClassifier(n_estimators=100)

# Fit the model to the training data
model_1.fit(X_train_processed, y_train_labels)
y_predict = model_1.predict(preprocessor(X_test))



## Define a Random Forest Model here, fit it, and score it



# Your cell should have a score between 0-1 as output
```

[36]:

```
#Example of self scoring by splitting training data
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score



X_train_split3, X_val3, y_train_split3, y_val3 = train_test_split(X_train,␣
 ↪y_train_labels, test_size=0.2, random_state=42)
model_3 = RandomForestClassifier(n_estimators=200, max_depth=5,␣
 ↪max_features='auto')
# Fit the model to the training data
X_train_processed3 = preprocessor(X_train_split3)
model_3.fit(X_train_processed3, y_train_split3)
y_predict3 = model_3.predict(preprocessor(X_val3))

score3 = accuracy_score(y_val3, y_predict3)
print("Model 3 score:", score3)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`
or remove this parameter as it is also the default value for
RandomForestClassifiers and ExtraTreesClassifiers.
```

```
  warn(
Model 3 score: 0.3888888888888889
```

Save preprocessor function to local "preprocessor.zip" file

[37]:
```python
import aimodelshare as ai
ai.export_preprocessor(preprocessor,"")
```

```
Warning: Please install pyspark to enable pyspark features
Your preprocessor is now saved to 'preprocessor.zip'
```

Save model to local ".onnx" file

[38]:
```python
# Save sklearn model to local ONNX file
from aimodelshare.aimsonnx import model_to_onnx

# Check how many preprocessed input features are there?
from skl2onnx.common.data_types import FloatTensorType

feature_count=preprocessor(X_test).shape[1] #Get count of preprocessed features
initial_type = [('float_input', FloatTensorType([None, feature_count]))] ␣
 ↪#Insert correct number of preprocessed features

onnx_model = model_to_onnx(model_1, framework='sklearn',
                          initial_types=initial_type,
                          transfer_learning=False,
                          deep_learning=False)

with open("model.onnx", "wb") as f:
    f.write(onnx_model.SerializeToString())
```

6.1 4. Generate predictions from X_test data and submit model to competition

[39]:
```python
#Set credentials using modelshare.org username/password

from aimodelshare.aws import set_credentials

#This is the unique rest api that powers this World Happiness Classification␣
 ↪Playground
## Do not change this
apiurl="https://e2w6gh3id1.execute-api.us-east-2.amazonaws.com/prod/m"

set_credentials(apiurl=apiurl)
```

```
Modelshare.ai Username:··········
Modelshare.ai Password:··········
Modelshare.ai login credentials set successfully.
```

[40]:
```
#Instantiate Competition
import aimodelshare as ai
mycompetition= ai.Competition(apiurl)
```

[41]:
```
#Submit Model 1:

#-- Generate predicted values (Model 1)
prediction_labels = model_1.predict(preprocessor(X_test))

model_filepath1 = "model.onnx"
preprocessor_filepath="preprocessor.zip"

# Submit Model 1 to Competition Leaderboard
mycompetition.submit_model(model = model_filepath1,
                           preprocessor=preprocessor_filepath,
                                 prediction_submission=prediction_labels)
```

```
Insert search tags to help users find your model (optional): QMSS
Provide any useful notes about your model (optional): Qiankun Li

Your model has been submitted as model version 1129

To submit code used to create this model or to view current leaderboard navigate
to Model Playground:

 https://www.modelshare.ai/detail/model:3164
```

[42]:
```
# Get leaderboard to explore current best model architectures

# Get raw data in pandas data frame
data = mycompetition.get_leaderboard()

# Stylize leaderboard data
mycompetition.stylize_leaderboard(data)
```

[42]:
```
<pandas.io.formats.style.Styler at 0x7bf124d88c70>
```

6.2 5. Repeat submission process to improve place on leaderboard

[43]:
```
# Train and submit model 2 using same preprocessor (note that you could save a␣
 ↪new preprocessor, but we will use the same one for this example).
from sklearn.ensemble import RandomForestClassifier

model_2 = RandomForestClassifier(n_estimators=200, max_depth = 5,␣
 ↪max_features='auto')
```

```
# Fit the model to the training data
model_2.fit(X_train_processed, y_train_labels)
y_predict = model_2.predict(preprocessor(X_test))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`
or remove this parameter as it is also the default value for
RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
```

Illustation of how the grading works

```
y_val_real = [
    'Low',
    'Average',
    'Low',
    'Very Low',
    'Very Low',
    'High',
    'Average',
    'High',
    'High',
    'Very Low',
    'Very High',
    'Average',
    'Very Low',
    'Low',
    'Low',
    'High',
    'High',
    'Average'
]


count = 0
for i in range(len(y_predict)):
  if y_predict[i] == y_val_real[i]:
    count += 1

print(count/len(y_val))
```

What changes did you make, what do the parameters you changed control, and why does it improve performance?

[44]:
```

```

```
## Your answer: I increase the number of estimators and the max depth. Both␣
↪parameters increase the complexity of the model, helping intepreting␣
↪complicated relations. It improves performance because increased number of␣
↪estimators and depth resolves underfitting.
```

[45]:
```
# Save sklearn model to local ONNX file
from aimodelshare.aimsonnx import model_to_onnx

feature_count= len(X_train.columns)# Get count of preprocessed features
initial_type = [('input', FloatTensorType([1, feature_count]))] # Insert␣
↪correct number of preprocessed features
onnx_model = model_to_onnx(model_2, framework='sklearn',
                          initial_types=initial_type,
                          transfer_learning=False,
                          deep_learning=False)

# Serialize your model to save it as an onnx file
```

[46]:
```
#Submit Model 2:

#-- Generate predicted y values (Model 2)
prediction_labels2 = model_2.predict(preprocessor(X_test))# Predict
model_filepath2 = "model.onnx"# Your serialized model

# Submit Model 2 to Competition Leaderboard
mycompetition.submit_model(model = model_filepath2,
                                 prediction_submission=prediction_labels2,
                                 preprocessor=preprocessor_filepath)
```

```
Insert search tags to help users find your model (optional): QMSS
Provide any useful notes about your model (optional): Qiankun Li

Your model has been submitted as model version 1130

To submit code used to create this model or to view current leaderboard navigate
to Model Playground:

 https://www.modelshare.ai/detail/model:3164
```

[47]:
```
# Compare two or more models
data=mycompetition.compare_models([1105,1108], verbose=1)
mycompetition.stylize_compare(data)
```

```
<IPython.core.display.HTML object>
```

Do you think it is worth making more changes to the parameters? Should we keep trying random values and see what works better? What is an alternative to doing this manually?

[48]:
```
## Your answer:It is no worth it, because adjusting by hand is too much work.␣
↪We need a systematic way (hyperparameter tunning) to computed the optimal␣
↪parameters. We should apply the grid hyperparameter search method to compare␣
↪different parameters. We try all possible combinations of paramters in␣
↪discrete space and use the parameters doing the best job.
```

[49]:
```
# Submit a third model using GridSearchCV

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
import numpy as np

param_grid = {
    'n_estimators': np.arange(100, 300, 100),  # Number of trees in the forest
    'max_depth': np.arange(5, 35, 10),  # Maximum depth of the tree
    'min_samples_split': np.arange(2, 10, 2),  # Minimum number of samples␣
↪required to split an internal node
    'min_samples_leaf': np.arange(1, 3),  # Minimum number of samples required␣
↪to be at a leaf node
    'max_features': ['auto', 'sqrt', 'log2']  # Number of features to consider␣
↪at every split
}# Use np.arange to create a sequence of numbers for each parameter's space you␣
↪think should be searched
rf = RandomForestClassifier()

gridmodel = GridSearchCV(rf, param_grid, cv=5, n_jobs=-1) # Read GridSearchCV␣
↪docs and create an object with RandomForestClassifier as the model

#use model methods to fit score and predict model:

# Fit the model to the training data
gridmodel.fit(X_train_processed, y_train_labels)


#extract best score and parameter by calling objects "best_score_" and␣
↪"best_params_"
print("best mean cross-validation score: {:.3f}".format(gridmodel.best_score_))
print("best parameters: {}".format(gridmodel.best_params_))
```

```
best mean cross-validation score: 0.672
best parameters: {'max_depth': 15, 'max_features': 'auto', 'min_samples_leaf':
```

```
1, 'min_samples_split': 2, 'n_estimators': 100}
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'`
or remove this parameter as it is also the default value for
RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
```

output of this cell is: best mean cross-validation score: 0.660 best parameters: {'max_depth': 5, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100} /usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also the default value for RandomForestClassifiers and ExtraTreesClassifiers. warn(

[50]:
```python
# Save sklearn model to local ONNX file
from aimodelshare.aimsonnx import model_to_onnx

feature_count=preprocessor(X_test).shape[1] #Get count of preprocessed features
initial_type = [('float_input', FloatTensorType([None, feature_count]))]  #␣
 ↪Insert correct number of preprocessed features

onnx_model = model_to_onnx(gridmodel, framework='sklearn',
                          initial_types=initial_type,
                          transfer_learning=False,
                          deep_learning=False)

with open("gridmodel.onnx", "wb") as f:
    f.write(onnx_model.SerializeToString())
```

[51]:
```python
#Submit Model 3:

#-- Generate predicted values
model_best = RandomForestClassifier(n_estimators=100, max_depth = 5,␣
 ↪min_samples_leaf = 1, min_samples_split = 4, max_features='log2')

# Fit the model to the training data
model_best.fit(X_train_processed, y_train_labels)

# Submit to Competition Leaderboard
prediction_labels_best = model_best.predict(preprocessor(X_test))# Predict
model_filepath4 = "model.onnx"# Your serialized model

# Submit Model 2 to Competition Leaderboard
mycompetition.submit_model(model = model_filepath4,
                                 prediction_submission=prediction_labels_best,
                                 preprocessor=preprocessor_filepath)
```

```
Insert search tags to help users find your model (optional): QMSS
Provide any useful notes about your model (optional): Qiankun Li

Your model has been submitted as model version 1131

To submit code used to create this model or to view current leaderboard navigate
to Model Playground:

 https://www.modelshare.ai/detail/model:3164
```

[52]:
```
# Get leaderboard

data = mycompetition.get_leaderboard()
mycompetition.stylize_leaderboard(data)
```

[52]:
```
<pandas.io.formats.style.Styler at 0x7bf02486fbe0>
```

[53]:
```
# Compare two or more models
data=mycompetition.compare_models([3,4], verbose=1)
mycompetition.stylize_compare(data)
```

```
<IPython.core.display.HTML object>
```

```
# Here are several classic ML architectures you can consider choosing from to experiment with
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier


model = ## Read documentations of imported models and fit them.

# Save sklearn model to local ONNX file
from aimodelshare.aimsonnx import model_to_onnx

feature_count=preprocessor(X_test).shape[1] #Get count of preprocessed features
initial_type = [('float_input', FloatTensorType([None, feature_count]))]  # Insert correct num

onnx_model = model_to_onnx(model, framework='sklearn',
                          initial_types=initial_type,
                          transfer_learning=False,
                          deep_learning=False)

with open("model.onnx", "wb") as f:
    f.write(onnx_model.SerializeToString())

#-- Generate predicted values
prediction_labels = model.predict(preprocessor(X_test))

model_filepath4 = "model.onnx"

# Submit model to Competition Leaderboard
mycompetition.submit_model(model = model_filepath4,
                                 prediction_submission=prediction_labels,
                                 preprocessor=preprocessor_filepath)
```

[69]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier


model4 = GradientBoostingClassifier(loss='log_loss',
                                    n_estimators=100,
                                    max_depth=5,
                                    learning_rate=0.1)

model4.fit(X_train_processed,y_train_labels)


with open("model.onnx", "wb") as f:
    f.write(onnx_model.SerializeToString())

# -- Generate predicted values
prediction_labels = model4.predict(preprocessor(X_test))

model_filepath4 = "model.onnx"

# Submit model to Competition Leaderboard
mycompetition.submit_model(model = model_filepath4,
                                 prediction_submission=prediction_labels,
                                 preprocessor=preprocessor_filepath)
```

```
Insert search tags to help users find your model (optional): QMSS
Provide any useful notes about your model (optional): Qiankun Li

Your model has been submitted as model version 1133

To submit code used to create this model or to view current leaderboard navigate
to Model Playground:
```

```
https://www.modelshare.ai/detail/model:3164
```

Describe what were the parameters you defined in GradientBoostingClassifier, and/or BaggingClassifier, and/or KNNs, and/or SVC? What worked and why?

[70]:
```
## Your answer: I defined the loss function to be log-loss. Also I defined the␣
↪number of estimators and max_depth, as well as the learning rate(rate of␣
↪descent when optimizing the loss function).
```

7 6. Basic Deep Learning

[71]:
```
feature_count = len(X_train.columns)
print(feature_count)
```

```
17
```

[72]:
```
# Now experiment with deep learning models:
import keras
from keras.models import Sequential
from keras.layers import Dense, Activation
import numpy as np


feature_count=X_train_processed.shape[1]#count features in input data

keras_model = Sequential()
keras_model.add(Dense(128, input_dim=feature_count, activation='relu'))
keras_model.add(Dense(64, activation='relu'))
keras_model.add(Dense(64, activation='relu'))
keras_model.add(Dense(32, activation='relu'))## Define a Neural Network Model␣
↪with 5 layers 128->64->64->32->(?)


#Use Softmax activation in last layer. How many neurons should there be in the␣
↪last layer?
num_classes = 5
keras_model.add(Dense(num_classes, activation='softmax'))


# Compile model
keras_model.compile(loss='categorical_crossentropy', optimizer='sgd',␣
↪metrics=['accuracy'])

# Fitting the NN to the Training set
keras_model.fit(preprocessor(X_train), y_train, ## Note that keras models␣
↪require a one-hot-encoded y_train object
               batch_size = 20,
                epochs = 300, validation_split=0.25)
```

```
Epoch 1/300
4/4 [==============================] - 1s 131ms/step - loss: 1.6623 - accuracy:
0.1818 - val_loss: 1.6515 - val_accuracy: 0.1818
Epoch 2/300
4/4 [==============================] - 0s 23ms/step - loss: 1.6410 - accuracy:
0.1970 - val_loss: 1.6397 - val_accuracy: 0.1818
Epoch 3/300
4/4 [==============================] - 0s 24ms/step - loss: 1.6225 - accuracy:
0.2424 - val_loss: 1.6277 - val_accuracy: 0.2273
Epoch 4/300
4/4 [==============================] - 0s 25ms/step - loss: 1.6038 - accuracy:
0.2576 - val_loss: 1.6133 - val_accuracy: 0.2727
Epoch 5/300
4/4 [==============================] - 0s 24ms/step - loss: 1.5828 - accuracy:
0.2727 - val_loss: 1.5986 - val_accuracy: 0.2727
Epoch 6/300
4/4 [==============================] - 0s 23ms/step - loss: 1.5640 - accuracy:
0.2879 - val_loss: 1.5848 - val_accuracy: 0.2727
Epoch 7/300
4/4 [==============================] - 0s 21ms/step - loss: 1.5444 - accuracy:
0.3485 - val_loss: 1.5712 - val_accuracy: 0.2273
Epoch 8/300
4/4 [==============================] - 0s 24ms/step - loss: 1.5249 - accuracy:
0.4545 - val_loss: 1.5633 - val_accuracy: 0.3182
Epoch 9/300
4/4 [==============================] - 0s 24ms/step - loss: 1.5097 - accuracy:
0.4848 - val_loss: 1.5537 - val_accuracy: 0.4091
Epoch 10/300
4/4 [==============================] - 0s 24ms/step - loss: 1.4947 - accuracy:
0.4848 - val_loss: 1.5433 - val_accuracy: 0.3636
Epoch 11/300
4/4 [==============================] - 0s 24ms/step - loss: 1.4792 - accuracy:
0.4697 - val_loss: 1.5340 - val_accuracy: 0.3182
Epoch 12/300
4/4 [==============================] - 0s 24ms/step - loss: 1.4649 - accuracy:
0.4848 - val_loss: 1.5216 - val_accuracy: 0.3182
Epoch 13/300
4/4 [==============================] - 0s 25ms/step - loss: 1.4467 - accuracy:
0.4848 - val_loss: 1.5120 - val_accuracy: 0.3182
Epoch 14/300
4/4 [==============================] - 0s 27ms/step - loss: 1.4334 - accuracy:
0.4848 - val_loss: 1.5036 - val_accuracy: 0.3182
Epoch 15/300
4/4 [==============================] - 0s 27ms/step - loss: 1.4187 - accuracy:
0.4697 - val_loss: 1.4929 - val_accuracy: 0.3636
```

```
Epoch 16/300
4/4 [==============================] - 0s 24ms/step - loss: 1.4041 - accuracy:
0.5000 - val_loss: 1.4858 - val_accuracy: 0.3636
Epoch 17/300
4/4 [==============================] - 0s 26ms/step - loss: 1.3894 - accuracy:
0.5000 - val_loss: 1.4788 - val_accuracy: 0.3636
Epoch 18/300
4/4 [==============================] - 0s 27ms/step - loss: 1.3772 - accuracy:
0.5303 - val_loss: 1.4674 - val_accuracy: 0.3636
Epoch 19/300
4/4 [==============================] - 0s 25ms/step - loss: 1.3605 - accuracy:
0.5000 - val_loss: 1.4584 - val_accuracy: 0.3636
Epoch 20/300
4/4 [==============================] - 0s 28ms/step - loss: 1.3469 - accuracy:
0.5152 - val_loss: 1.4517 - val_accuracy: 0.3636
Epoch 21/300
4/4 [==============================] - 0s 23ms/step - loss: 1.3330 - accuracy:
0.5303 - val_loss: 1.4418 - val_accuracy: 0.4091
Epoch 22/300
4/4 [==============================] - 0s 27ms/step - loss: 1.3192 - accuracy:
0.5455 - val_loss: 1.4317 - val_accuracy: 0.4091
Epoch 23/300
4/4 [==============================] - 0s 31ms/step - loss: 1.3057 - accuracy:
0.5606 - val_loss: 1.4214 - val_accuracy: 0.4091
Epoch 24/300
4/4 [==============================] - 0s 28ms/step - loss: 1.2910 - accuracy:
0.5455 - val_loss: 1.4143 - val_accuracy: 0.4091
Epoch 25/300
4/4 [==============================] - 0s 24ms/step - loss: 1.2791 - accuracy:
0.5606 - val_loss: 1.4056 - val_accuracy: 0.4091
Epoch 26/300
4/4 [==============================] - 0s 25ms/step - loss: 1.2668 - accuracy:
0.5606 - val_loss: 1.3983 - val_accuracy: 0.4545
Epoch 27/300
4/4 [==============================] - 0s 20ms/step - loss: 1.2524 - accuracy:
0.5606 - val_loss: 1.3911 - val_accuracy: 0.4545
Epoch 28/300
4/4 [==============================] - 0s 19ms/step - loss: 1.2393 - accuracy:
0.5606 - val_loss: 1.3818 - val_accuracy: 0.4091
Epoch 29/300
4/4 [==============================] - 0s 20ms/step - loss: 1.2280 - accuracy:
0.5909 - val_loss: 1.3742 - val_accuracy: 0.4545
Epoch 30/300
4/4 [==============================] - 0s 14ms/step - loss: 1.2136 - accuracy:
0.5909 - val_loss: 1.3668 - val_accuracy: 0.4545
Epoch 31/300
4/4 [==============================] - 0s 14ms/step - loss: 1.2007 - accuracy:
0.5909 - val_loss: 1.3592 - val_accuracy: 0.4545
```

```
Epoch 32/300
4/4 [==============================] - 0s 21ms/step - loss: 1.1881 - accuracy:
0.5909 - val_loss: 1.3538 - val_accuracy: 0.5000
Epoch 33/300
4/4 [==============================] - 0s 15ms/step - loss: 1.1771 - accuracy:
0.6212 - val_loss: 1.3467 - val_accuracy: 0.4545
Epoch 34/300
4/4 [==============================] - 0s 21ms/step - loss: 1.1644 - accuracy:
0.6212 - val_loss: 1.3443 - val_accuracy: 0.4545
Epoch 35/300
4/4 [==============================] - 0s 20ms/step - loss: 1.1545 - accuracy:
0.6212 - val_loss: 1.3398 - val_accuracy: 0.4545
Epoch 36/300
4/4 [==============================] - 0s 13ms/step - loss: 1.1436 - accuracy:
0.6364 - val_loss: 1.3312 - val_accuracy: 0.5000
Epoch 37/300
4/4 [==============================] - 0s 19ms/step - loss: 1.1326 - accuracy:
0.6212 - val_loss: 1.3249 - val_accuracy: 0.5000
Epoch 38/300
4/4 [==============================] - 0s 14ms/step - loss: 1.1217 - accuracy:
0.6212 - val_loss: 1.3177 - val_accuracy: 0.5000
Epoch 39/300
4/4 [==============================] - 0s 19ms/step - loss: 1.1098 - accuracy:
0.6212 - val_loss: 1.3113 - val_accuracy: 0.4545
Epoch 40/300
4/4 [==============================] - 0s 19ms/step - loss: 1.1007 - accuracy:
0.6212 - val_loss: 1.3021 - val_accuracy: 0.4545
Epoch 41/300
4/4 [==============================] - 0s 15ms/step - loss: 1.0904 - accuracy:
0.6212 - val_loss: 1.2934 - val_accuracy: 0.4545
Epoch 42/300
4/4 [==============================] - 0s 15ms/step - loss: 1.0790 - accuracy:
0.6212 - val_loss: 1.2895 - val_accuracy: 0.4545
Epoch 43/300
4/4 [==============================] - 0s 19ms/step - loss: 1.0702 - accuracy:
0.6212 - val_loss: 1.2793 - val_accuracy: 0.4545
Epoch 44/300
4/4 [==============================] - 0s 14ms/step - loss: 1.0615 - accuracy:
0.6212 - val_loss: 1.2722 - val_accuracy: 0.4545
Epoch 45/300
4/4 [==============================] - 0s 20ms/step - loss: 1.0531 - accuracy:
0.6212 - val_loss: 1.2666 - val_accuracy: 0.4545
Epoch 46/300
4/4 [==============================] - 0s 18ms/step - loss: 1.0432 - accuracy:
0.6212 - val_loss: 1.2647 - val_accuracy: 0.4545
Epoch 47/300
4/4 [==============================] - 0s 19ms/step - loss: 1.0339 - accuracy:
0.6667 - val_loss: 1.2619 - val_accuracy: 0.4545
```

```
Epoch 48/300
4/4 [==============================] - 0s 13ms/step - loss: 1.0255 - accuracy:
0.6667 - val_loss: 1.2542 - val_accuracy: 0.4545
Epoch 49/300
4/4 [==============================] - 0s 13ms/step - loss: 1.0152 - accuracy:
0.6667 - val_loss: 1.2472 - val_accuracy: 0.4545
Epoch 50/300
4/4 [==============================] - 0s 22ms/step - loss: 1.0074 - accuracy:
0.6818 - val_loss: 1.2442 - val_accuracy: 0.4545
Epoch 51/300
4/4 [==============================] - 0s 19ms/step - loss: 1.0006 - accuracy:
0.6667 - val_loss: 1.2415 - val_accuracy: 0.4545
Epoch 52/300
4/4 [==============================] - 0s 20ms/step - loss: 0.9916 - accuracy:
0.6667 - val_loss: 1.2405 - val_accuracy: 0.4545
Epoch 53/300
4/4 [==============================] - 0s 19ms/step - loss: 0.9811 - accuracy:
0.6818 - val_loss: 1.2338 - val_accuracy: 0.4545
Epoch 54/300
4/4 [==============================] - 0s 19ms/step - loss: 0.9732 - accuracy:
0.6818 - val_loss: 1.2239 - val_accuracy: 0.4545
Epoch 55/300
4/4 [==============================] - 0s 18ms/step - loss: 0.9648 - accuracy:
0.6818 - val_loss: 1.2160 - val_accuracy: 0.4545
Epoch 56/300
4/4 [==============================] - 0s 13ms/step - loss: 0.9562 - accuracy:
0.6818 - val_loss: 1.2068 - val_accuracy: 0.4545
Epoch 57/300
4/4 [==============================] - 0s 15ms/step - loss: 0.9496 - accuracy:
0.6818 - val_loss: 1.2036 - val_accuracy: 0.4545
Epoch 58/300
4/4 [==============================] - 0s 14ms/step - loss: 0.9416 - accuracy:
0.6818 - val_loss: 1.1945 - val_accuracy: 0.4545
Epoch 59/300
4/4 [==============================] - 0s 18ms/step - loss: 0.9329 - accuracy:
0.6818 - val_loss: 1.1975 - val_accuracy: 0.4545
Epoch 60/300
4/4 [==============================] - 0s 14ms/step - loss: 0.9267 - accuracy:
0.6818 - val_loss: 1.1830 - val_accuracy: 0.4545
Epoch 61/300
4/4 [==============================] - 0s 13ms/step - loss: 0.9164 - accuracy:
0.6970 - val_loss: 1.1790 - val_accuracy: 0.4545
Epoch 62/300
4/4 [==============================] - 0s 14ms/step - loss: 0.9099 - accuracy:
0.6970 - val_loss: 1.1684 - val_accuracy: 0.4545
Epoch 63/300
4/4 [==============================] - 0s 19ms/step - loss: 0.9030 - accuracy:
0.6970 - val_loss: 1.1635 - val_accuracy: 0.5000
```

```
Epoch 64/300
4/4 [==============================] - 0s 19ms/step - loss: 0.8958 - accuracy:
0.6970 - val_loss: 1.1601 - val_accuracy: 0.5455
Epoch 65/300
4/4 [==============================] - 0s 15ms/step - loss: 0.8896 - accuracy:
0.6970 - val_loss: 1.1496 - val_accuracy: 0.5455
Epoch 66/300
4/4 [==============================] - 0s 19ms/step - loss: 0.8834 - accuracy:
0.6970 - val_loss: 1.1478 - val_accuracy: 0.5000
Epoch 67/300
4/4 [==============================] - 0s 19ms/step - loss: 0.8743 - accuracy:
0.7121 - val_loss: 1.1488 - val_accuracy: 0.5000
Epoch 68/300
4/4 [==============================] - 0s 14ms/step - loss: 0.8671 - accuracy:
0.7121 - val_loss: 1.1417 - val_accuracy: 0.5455
Epoch 69/300
4/4 [==============================] - 0s 20ms/step - loss: 0.8595 - accuracy:
0.7121 - val_loss: 1.1396 - val_accuracy: 0.5000
Epoch 70/300
4/4 [==============================] - 0s 20ms/step - loss: 0.8529 - accuracy:
0.7121 - val_loss: 1.1332 - val_accuracy: 0.5455
Epoch 71/300
4/4 [==============================] - 0s 20ms/step - loss: 0.8452 - accuracy:
0.7121 - val_loss: 1.1230 - val_accuracy: 0.5455
Epoch 72/300
4/4 [==============================] - 0s 16ms/step - loss: 0.8380 - accuracy:
0.7121 - val_loss: 1.1166 - val_accuracy: 0.5455
Epoch 73/300
4/4 [==============================] - 0s 14ms/step - loss: 0.8303 - accuracy:
0.7121 - val_loss: 1.1162 - val_accuracy: 0.5455
Epoch 74/300
4/4 [==============================] - 0s 20ms/step - loss: 0.8234 - accuracy:
0.7121 - val_loss: 1.1152 - val_accuracy: 0.5455
Epoch 75/300
4/4 [==============================] - 0s 14ms/step - loss: 0.8172 - accuracy:
0.7121 - val_loss: 1.1143 - val_accuracy: 0.5455
Epoch 76/300
4/4 [==============================] - 0s 18ms/step - loss: 0.8122 - accuracy:
0.7121 - val_loss: 1.1100 - val_accuracy: 0.5455
Epoch 77/300
4/4 [==============================] - 0s 14ms/step - loss: 0.8039 - accuracy:
0.7424 - val_loss: 1.1064 - val_accuracy: 0.5455
Epoch 78/300
4/4 [==============================] - 0s 15ms/step - loss: 0.7985 - accuracy:
0.7424 - val_loss: 1.1043 - val_accuracy: 0.5455
Epoch 79/300
4/4 [==============================] - 0s 15ms/step - loss: 0.7933 - accuracy:
0.7121 - val_loss: 1.0945 - val_accuracy: 0.5455
```

```
Epoch 80/300
4/4 [==============================] - 0s 14ms/step - loss: 0.7869 - accuracy:
0.7273 - val_loss: 1.0851 - val_accuracy: 0.5455
Epoch 81/300
4/4 [==============================] - 0s 19ms/step - loss: 0.7841 - accuracy:
0.7576 - val_loss: 1.0728 - val_accuracy: 0.5909
Epoch 82/300
4/4 [==============================] - 0s 13ms/step - loss: 0.7757 - accuracy:
0.7879 - val_loss: 1.0624 - val_accuracy: 0.5909
Epoch 83/300
4/4 [==============================] - 0s 24ms/step - loss: 0.7691 - accuracy:
0.7879 - val_loss: 1.0631 - val_accuracy: 0.5909
Epoch 84/300
4/4 [==============================] - 0s 14ms/step - loss: 0.7614 - accuracy:
0.7879 - val_loss: 1.0683 - val_accuracy: 0.5909
Epoch 85/300
4/4 [==============================] - 0s 21ms/step - loss: 0.7555 - accuracy:
0.7727 - val_loss: 1.0646 - val_accuracy: 0.5909
Epoch 86/300
4/4 [==============================] - 0s 15ms/step - loss: 0.7495 - accuracy:
0.7727 - val_loss: 1.0660 - val_accuracy: 0.5455
Epoch 87/300
4/4 [==============================] - 0s 15ms/step - loss: 0.7419 - accuracy:
0.7727 - val_loss: 1.0691 - val_accuracy: 0.5455
Epoch 88/300
4/4 [==============================] - 0s 15ms/step - loss: 0.7367 - accuracy:
0.7727 - val_loss: 1.0680 - val_accuracy: 0.5455
Epoch 89/300
4/4 [==============================] - 0s 19ms/step - loss: 0.7329 - accuracy:
0.7727 - val_loss: 1.0494 - val_accuracy: 0.5455
Epoch 90/300
4/4 [==============================] - 0s 14ms/step - loss: 0.7267 - accuracy:
0.7879 - val_loss: 1.0419 - val_accuracy: 0.5455
Epoch 91/300
4/4 [==============================] - 0s 19ms/step - loss: 0.7203 - accuracy:
0.7879 - val_loss: 1.0393 - val_accuracy: 0.5455
Epoch 92/300
4/4 [==============================] - 0s 15ms/step - loss: 0.7152 - accuracy:
0.7879 - val_loss: 1.0553 - val_accuracy: 0.5455
Epoch 93/300
4/4 [==============================] - 0s 19ms/step - loss: 0.7113 - accuracy:
0.7727 - val_loss: 1.0485 - val_accuracy: 0.5455
Epoch 94/300
4/4 [==============================] - 0s 20ms/step - loss: 0.7074 - accuracy:
0.7727 - val_loss: 1.0620 - val_accuracy: 0.5000
Epoch 95/300
4/4 [==============================] - 0s 14ms/step - loss: 0.7027 - accuracy:
0.7727 - val_loss: 1.0403 - val_accuracy: 0.5455
```

```
Epoch 96/300
4/4 [==============================] - 0s 20ms/step - loss: 0.6954 - accuracy:
0.7879 - val_loss: 1.0457 - val_accuracy: 0.5455
Epoch 97/300
4/4 [==============================] - 0s 14ms/step - loss: 0.6905 - accuracy:
0.7879 - val_loss: 1.0350 - val_accuracy: 0.5455
Epoch 98/300
4/4 [==============================] - 0s 19ms/step - loss: 0.6842 - accuracy:
0.7879 - val_loss: 1.0253 - val_accuracy: 0.5455
Epoch 99/300
4/4 [==============================] - 0s 21ms/step - loss: 0.6767 - accuracy:
0.8030 - val_loss: 1.0234 - val_accuracy: 0.5455
Epoch 100/300
4/4 [==============================] - 0s 14ms/step - loss: 0.6727 - accuracy:
0.7879 - val_loss: 1.0261 - val_accuracy: 0.5455
Epoch 101/300
4/4 [==============================] - 0s 15ms/step - loss: 0.6671 - accuracy:
0.7879 - val_loss: 1.0197 - val_accuracy: 0.5455
Epoch 102/300
4/4 [==============================] - 0s 21ms/step - loss: 0.6658 - accuracy:
0.8030 - val_loss: 1.0246 - val_accuracy: 0.5455
Epoch 103/300
4/4 [==============================] - 0s 33ms/step - loss: 0.6606 - accuracy:
0.8030 - val_loss: 1.0088 - val_accuracy: 0.5455
Epoch 104/300
4/4 [==============================] - 0s 48ms/step - loss: 0.6575 - accuracy:
0.8030 - val_loss: 1.0064 - val_accuracy: 0.5455
Epoch 105/300
4/4 [==============================] - 0s 41ms/step - loss: 0.6495 - accuracy:
0.8030 - val_loss: 1.0088 - val_accuracy: 0.5455
Epoch 106/300
4/4 [==============================] - 0s 59ms/step - loss: 0.6448 - accuracy:
0.7879 - val_loss: 1.0098 - val_accuracy: 0.5455
Epoch 107/300
4/4 [==============================] - 0s 72ms/step - loss: 0.6393 - accuracy:
0.8030 - val_loss: 1.0023 - val_accuracy: 0.5455
Epoch 108/300
4/4 [==============================] - 0s 24ms/step - loss: 0.6349 - accuracy:
0.8030 - val_loss: 1.0060 - val_accuracy: 0.5455
Epoch 109/300
4/4 [==============================] - 0s 57ms/step - loss: 0.6328 - accuracy:
0.7879 - val_loss: 1.0069 - val_accuracy: 0.5455
Epoch 110/300
4/4 [==============================] - 0s 13ms/step - loss: 0.6279 - accuracy:
0.7879 - val_loss: 1.0089 - val_accuracy: 0.5455
Epoch 111/300
4/4 [==============================] - 0s 19ms/step - loss: 0.6216 - accuracy:
0.7879 - val_loss: 1.0162 - val_accuracy: 0.5455
```

```
Epoch 112/300
4/4 [==============================] - 0s 19ms/step - loss: 0.6183 - accuracy:
0.7879 - val_loss: 1.0200 - val_accuracy: 0.5455
Epoch 113/300
4/4 [==============================] - 0s 19ms/step - loss: 0.6149 - accuracy:
0.7727 - val_loss: 1.0028 - val_accuracy: 0.5455
Epoch 114/300
4/4 [==============================] - 0s 15ms/step - loss: 0.6085 - accuracy:
0.7879 - val_loss: 1.0058 - val_accuracy: 0.5455
Epoch 115/300
4/4 [==============================] - 0s 14ms/step - loss: 0.6043 - accuracy:
0.7879 - val_loss: 1.0078 - val_accuracy: 0.5455
Epoch 116/300
4/4 [==============================] - 0s 20ms/step - loss: 0.6006 - accuracy:
0.7879 - val_loss: 0.9954 - val_accuracy: 0.5909
Epoch 117/300
4/4 [==============================] - 0s 21ms/step - loss: 0.5978 - accuracy:
0.8030 - val_loss: 1.0050 - val_accuracy: 0.5455
Epoch 118/300
4/4 [==============================] - 0s 14ms/step - loss: 0.5904 - accuracy:
0.7879 - val_loss: 0.9899 - val_accuracy: 0.5455
Epoch 119/300
4/4 [==============================] - 0s 14ms/step - loss: 0.5857 - accuracy:
0.8030 - val_loss: 0.9832 - val_accuracy: 0.5455
Epoch 120/300
4/4 [==============================] - 0s 21ms/step - loss: 0.5807 - accuracy:
0.8030 - val_loss: 0.9915 - val_accuracy: 0.5455
Epoch 121/300
4/4 [==============================] - 0s 18ms/step - loss: 0.5778 - accuracy:
0.8030 - val_loss: 1.0035 - val_accuracy: 0.5909
Epoch 122/300
4/4 [==============================] - 0s 15ms/step - loss: 0.5769 - accuracy:
0.7879 - val_loss: 1.0117 - val_accuracy: 0.5909
Epoch 123/300
4/4 [==============================] - 0s 20ms/step - loss: 0.5669 - accuracy:
0.8182 - val_loss: 1.0083 - val_accuracy: 0.5909
Epoch 124/300
4/4 [==============================] - 0s 20ms/step - loss: 0.5673 - accuracy:
0.8182 - val_loss: 1.0282 - val_accuracy: 0.5455
Epoch 125/300
4/4 [==============================] - 0s 14ms/step - loss: 0.5649 - accuracy:
0.8182 - val_loss: 0.9974 - val_accuracy: 0.5909
Epoch 126/300
4/4 [==============================] - 0s 19ms/step - loss: 0.5614 - accuracy:
0.8333 - val_loss: 0.9916 - val_accuracy: 0.5909
Epoch 127/300
4/4 [==============================] - 0s 20ms/step - loss: 0.5557 - accuracy:
0.8333 - val_loss: 0.9831 - val_accuracy: 0.5909
```

```
Epoch 128/300
4/4 [==============================] - 0s 14ms/step - loss: 0.5493 - accuracy:
0.8333 - val_loss: 0.9790 - val_accuracy: 0.5909
Epoch 129/300
4/4 [==============================] - 0s 14ms/step - loss: 0.5485 - accuracy:
0.8636 - val_loss: 0.9890 - val_accuracy: 0.5909
Epoch 130/300
4/4 [==============================] - 0s 19ms/step - loss: 0.5411 - accuracy:
0.8485 - val_loss: 0.9876 - val_accuracy: 0.5909
Epoch 131/300
4/4 [==============================] - 0s 19ms/step - loss: 0.5397 - accuracy:
0.8333 - val_loss: 0.9764 - val_accuracy: 0.5909
Epoch 132/300
4/4 [==============================] - 0s 13ms/step - loss: 0.5363 - accuracy:
0.8485 - val_loss: 0.9888 - val_accuracy: 0.5909
Epoch 133/300
4/4 [==============================] - 0s 19ms/step - loss: 0.5331 - accuracy:
0.8333 - val_loss: 0.9941 - val_accuracy: 0.5909
Epoch 134/300
4/4 [==============================] - 0s 19ms/step - loss: 0.5286 - accuracy:
0.8182 - val_loss: 0.9736 - val_accuracy: 0.5909
Epoch 135/300
4/4 [==============================] - 0s 19ms/step - loss: 0.5275 - accuracy:
0.8485 - val_loss: 0.9857 - val_accuracy: 0.5909
Epoch 136/300
4/4 [==============================] - 0s 16ms/step - loss: 0.5196 - accuracy:
0.8485 - val_loss: 0.9888 - val_accuracy: 0.5909
Epoch 137/300
4/4 [==============================] - 0s 17ms/step - loss: 0.5159 - accuracy:
0.8485 - val_loss: 0.9809 - val_accuracy: 0.5909
Epoch 138/300
4/4 [==============================] - 0s 19ms/step - loss: 0.5183 - accuracy:
0.8636 - val_loss: 1.0206 - val_accuracy: 0.5455
Epoch 139/300
4/4 [==============================] - 0s 19ms/step - loss: 0.5149 - accuracy:
0.8485 - val_loss: 0.9944 - val_accuracy: 0.5909
Epoch 140/300
4/4 [==============================] - 0s 21ms/step - loss: 0.5047 - accuracy:
0.8485 - val_loss: 0.9789 - val_accuracy: 0.5909
Epoch 141/300
4/4 [==============================] - 0s 19ms/step - loss: 0.5033 - accuracy:
0.8485 - val_loss: 0.9810 - val_accuracy: 0.5909
Epoch 142/300
4/4 [==============================] - 0s 14ms/step - loss: 0.5034 - accuracy:
0.8485 - val_loss: 0.9922 - val_accuracy: 0.5909
Epoch 143/300
4/4 [==============================] - 0s 24ms/step - loss: 0.4977 - accuracy:
0.8485 - val_loss: 0.9639 - val_accuracy: 0.6364
```

```
Epoch 144/300
4/4 [==============================] - 0s 20ms/step - loss: 0.4965 - accuracy:
0.8636 - val_loss: 0.9641 - val_accuracy: 0.6364
Epoch 145/300
4/4 [==============================] - 0s 19ms/step - loss: 0.4919 - accuracy:
0.8788 - val_loss: 0.9587 - val_accuracy: 0.6364
Epoch 146/300
4/4 [==============================] - 0s 18ms/step - loss: 0.4959 - accuracy:
0.8636 - val_loss: 0.9650 - val_accuracy: 0.5909
Epoch 147/300
4/4 [==============================] - 0s 19ms/step - loss: 0.4861 - accuracy:
0.8788 - val_loss: 0.9664 - val_accuracy: 0.6364
Epoch 148/300
4/4 [==============================] - 0s 23ms/step - loss: 0.4823 - accuracy:
0.8788 - val_loss: 0.9782 - val_accuracy: 0.6364
Epoch 149/300
4/4 [==============================] - 0s 14ms/step - loss: 0.4786 - accuracy:
0.8485 - val_loss: 0.9963 - val_accuracy: 0.5909
Epoch 150/300
4/4 [==============================] - 0s 19ms/step - loss: 0.4754 - accuracy:
0.8636 - val_loss: 0.9884 - val_accuracy: 0.5909
Epoch 151/300
4/4 [==============================] - 0s 14ms/step - loss: 0.4722 - accuracy:
0.8485 - val_loss: 1.0107 - val_accuracy: 0.5909
Epoch 152/300
4/4 [==============================] - 0s 24ms/step - loss: 0.4698 - accuracy:
0.8788 - val_loss: 1.0095 - val_accuracy: 0.5455
Epoch 153/300
4/4 [==============================] - 0s 15ms/step - loss: 0.4691 - accuracy:
0.8636 - val_loss: 0.9963 - val_accuracy: 0.5909
Epoch 154/300
4/4 [==============================] - 0s 14ms/step - loss: 0.4599 - accuracy:
0.8636 - val_loss: 0.9978 - val_accuracy: 0.5909
Epoch 155/300
4/4 [==============================] - 0s 19ms/step - loss: 0.4567 - accuracy:
0.8636 - val_loss: 1.0063 - val_accuracy: 0.5909
Epoch 156/300
4/4 [==============================] - 0s 14ms/step - loss: 0.4555 - accuracy:
0.8636 - val_loss: 1.0049 - val_accuracy: 0.5909
Epoch 157/300
4/4 [==============================] - 0s 16ms/step - loss: 0.4523 - accuracy:
0.8636 - val_loss: 1.0029 - val_accuracy: 0.5909
Epoch 158/300
4/4 [==============================] - 0s 15ms/step - loss: 0.4525 - accuracy:
0.8485 - val_loss: 0.9822 - val_accuracy: 0.6364
Epoch 159/300
4/4 [==============================] - 0s 14ms/step - loss: 0.4507 - accuracy:
0.8636 - val_loss: 0.9666 - val_accuracy: 0.5909
```

```
Epoch 160/300
4/4 [==============================] - 0s 21ms/step - loss: 0.4476 - accuracy:
0.8636 - val_loss: 0.9853 - val_accuracy: 0.5909
Epoch 161/300
4/4 [==============================] - 0s 14ms/step - loss: 0.4417 - accuracy:
0.8485 - val_loss: 0.9911 - val_accuracy: 0.6364
Epoch 162/300
4/4 [==============================] - 0s 15ms/step - loss: 0.4430 - accuracy:
0.8485 - val_loss: 0.9889 - val_accuracy: 0.6364
Epoch 163/300
4/4 [==============================] - 0s 20ms/step - loss: 0.4384 - accuracy:
0.8333 - val_loss: 0.9941 - val_accuracy: 0.6364
Epoch 164/300
4/4 [==============================] - 0s 13ms/step - loss: 0.4307 - accuracy:
0.8636 - val_loss: 0.9981 - val_accuracy: 0.5909
Epoch 165/300
4/4 [==============================] - 0s 16ms/step - loss: 0.4298 - accuracy:
0.8485 - val_loss: 0.9961 - val_accuracy: 0.5909
Epoch 166/300
4/4 [==============================] - 0s 14ms/step - loss: 0.4280 - accuracy:
0.8788 - val_loss: 1.0047 - val_accuracy: 0.5909
Epoch 167/300
4/4 [==============================] - 0s 16ms/step - loss: 0.4272 - accuracy:
0.8636 - val_loss: 1.0160 - val_accuracy: 0.5909
Epoch 168/300
4/4 [==============================] - 0s 15ms/step - loss: 0.4228 - accuracy:
0.8636 - val_loss: 1.0021 - val_accuracy: 0.6364
Epoch 169/300
4/4 [==============================] - 0s 18ms/step - loss: 0.4188 - accuracy:
0.8788 - val_loss: 1.0197 - val_accuracy: 0.5909
Epoch 170/300
4/4 [==============================] - 0s 14ms/step - loss: 0.4151 - accuracy:
0.8788 - val_loss: 1.0068 - val_accuracy: 0.6364
Epoch 171/300
4/4 [==============================] - 0s 16ms/step - loss: 0.4094 - accuracy:
0.8788 - val_loss: 1.0063 - val_accuracy: 0.5909
Epoch 172/300
4/4 [==============================] - 0s 19ms/step - loss: 0.4073 - accuracy:
0.8788 - val_loss: 1.0171 - val_accuracy: 0.5455
Epoch 173/300
4/4 [==============================] - 0s 20ms/step - loss: 0.4060 - accuracy:
0.8788 - val_loss: 0.9943 - val_accuracy: 0.6364
Epoch 174/300
4/4 [==============================] - 0s 27ms/step - loss: 0.4042 - accuracy:
0.8788 - val_loss: 0.9965 - val_accuracy: 0.5909
Epoch 175/300
4/4 [==============================] - 0s 26ms/step - loss: 0.4011 - accuracy:
0.8939 - val_loss: 1.0056 - val_accuracy: 0.5909
```

```
Epoch 176/300
4/4 [==============================] - 0s 20ms/step - loss: 0.3952 - accuracy:
0.8788 - val_loss: 1.0148 - val_accuracy: 0.5909
Epoch 177/300
4/4 [==============================] - 0s 29ms/step - loss: 0.4007 - accuracy:
0.8636 - val_loss: 1.0267 - val_accuracy: 0.5909
Epoch 178/300
4/4 [==============================] - 0s 25ms/step - loss: 0.3920 - accuracy:
0.8788 - val_loss: 1.0395 - val_accuracy: 0.5909
Epoch 179/300
4/4 [==============================] - 0s 21ms/step - loss: 0.3902 - accuracy:
0.8636 - val_loss: 1.0601 - val_accuracy: 0.5455
Epoch 180/300
4/4 [==============================] - 0s 24ms/step - loss: 0.3876 - accuracy:
0.8939 - val_loss: 1.0242 - val_accuracy: 0.6364
Epoch 181/300
4/4 [==============================] - 0s 19ms/step - loss: 0.3884 - accuracy:
0.8788 - val_loss: 1.0467 - val_accuracy: 0.5909
Epoch 182/300
4/4 [==============================] - 0s 25ms/step - loss: 0.3825 - accuracy:
0.8939 - val_loss: 1.0633 - val_accuracy: 0.5455
Epoch 183/300
4/4 [==============================] - 0s 28ms/step - loss: 0.3779 - accuracy:
0.8788 - val_loss: 1.0653 - val_accuracy: 0.5455
Epoch 184/300
4/4 [==============================] - 0s 19ms/step - loss: 0.3761 - accuracy:
0.8636 - val_loss: 1.0587 - val_accuracy: 0.5455
Epoch 185/300
4/4 [==============================] - 0s 19ms/step - loss: 0.3719 - accuracy:
0.8788 - val_loss: 1.0478 - val_accuracy: 0.5909
Epoch 186/300
4/4 [==============================] - 0s 24ms/step - loss: 0.3729 - accuracy:
0.8939 - val_loss: 1.0407 - val_accuracy: 0.5909
Epoch 187/300
4/4 [==============================] - 0s 27ms/step - loss: 0.3715 - accuracy:
0.8939 - val_loss: 1.0353 - val_accuracy: 0.5909
Epoch 188/300
4/4 [==============================] - 0s 21ms/step - loss: 0.3613 - accuracy:
0.8939 - val_loss: 1.0365 - val_accuracy: 0.5909
Epoch 189/300
4/4 [==============================] - 0s 26ms/step - loss: 0.3598 - accuracy:
0.9091 - val_loss: 1.1064 - val_accuracy: 0.5455
Epoch 190/300
4/4 [==============================] - 0s 25ms/step - loss: 0.3636 - accuracy:
0.9091 - val_loss: 1.0442 - val_accuracy: 0.5909
Epoch 191/300
4/4 [==============================] - 0s 23ms/step - loss: 0.3561 - accuracy:
0.8939 - val_loss: 1.0408 - val_accuracy: 0.5909
```

```
Epoch 192/300
4/4 [==============================] - 0s 26ms/step - loss: 0.3568 - accuracy:
0.8939 - val_loss: 1.0581 - val_accuracy: 0.5909
Epoch 193/300
4/4 [==============================] - 0s 21ms/step - loss: 0.3553 - accuracy:
0.8788 - val_loss: 1.0321 - val_accuracy: 0.6818
Epoch 194/300
4/4 [==============================] - 0s 27ms/step - loss: 0.3536 - accuracy:
0.8939 - val_loss: 1.0813 - val_accuracy: 0.5909
Epoch 195/300
4/4 [==============================] - 0s 33ms/step - loss: 0.3495 - accuracy:
0.9091 - val_loss: 1.0300 - val_accuracy: 0.6364
Epoch 196/300
4/4 [==============================] - 0s 24ms/step - loss: 0.3446 - accuracy:
0.9091 - val_loss: 1.0584 - val_accuracy: 0.5455
Epoch 197/300
4/4 [==============================] - 0s 26ms/step - loss: 0.3431 - accuracy:
0.9242 - val_loss: 1.0583 - val_accuracy: 0.5909
Epoch 198/300
4/4 [==============================] - 0s 26ms/step - loss: 0.3379 - accuracy:
0.8939 - val_loss: 1.0246 - val_accuracy: 0.6818
Epoch 199/300
4/4 [==============================] - 0s 24ms/step - loss: 0.3384 - accuracy:
0.9242 - val_loss: 1.0622 - val_accuracy: 0.5909
Epoch 200/300
4/4 [==============================] - 0s 25ms/step - loss: 0.3316 - accuracy:
0.9242 - val_loss: 1.0518 - val_accuracy: 0.5455
Epoch 201/300
4/4 [==============================] - 0s 24ms/step - loss: 0.3268 - accuracy:
0.9242 - val_loss: 1.0566 - val_accuracy: 0.6364
Epoch 202/300
4/4 [==============================] - 0s 25ms/step - loss: 0.3299 - accuracy:
0.9242 - val_loss: 1.0711 - val_accuracy: 0.5455
Epoch 203/300
4/4 [==============================] - 0s 27ms/step - loss: 0.3298 - accuracy:
0.9242 - val_loss: 1.0801 - val_accuracy: 0.5909
Epoch 204/300
4/4 [==============================] - 0s 28ms/step - loss: 0.3236 - accuracy:
0.9242 - val_loss: 1.0926 - val_accuracy: 0.5455
Epoch 205/300
4/4 [==============================] - 0s 24ms/step - loss: 0.3207 - accuracy:
0.9091 - val_loss: 1.0781 - val_accuracy: 0.5909
Epoch 206/300
4/4 [==============================] - 0s 20ms/step - loss: 0.3219 - accuracy:
0.9091 - val_loss: 1.0990 - val_accuracy: 0.5455
Epoch 207/300
4/4 [==============================] - 0s 25ms/step - loss: 0.3161 - accuracy:
0.9242 - val_loss: 1.0768 - val_accuracy: 0.5455
```

```
Epoch 208/300
4/4 [==============================] - 0s 14ms/step - loss: 0.3157 - accuracy:
0.9242 - val_loss: 1.0738 - val_accuracy: 0.6364
Epoch 209/300
4/4 [==============================] - 0s 20ms/step - loss: 0.3138 - accuracy:
0.9242 - val_loss: 1.0841 - val_accuracy: 0.5909
Epoch 210/300
4/4 [==============================] - 0s 15ms/step - loss: 0.3097 - accuracy:
0.9242 - val_loss: 1.0770 - val_accuracy: 0.6364
Epoch 211/300
4/4 [==============================] - 0s 15ms/step - loss: 0.3097 - accuracy:
0.9242 - val_loss: 1.1454 - val_accuracy: 0.5455
Epoch 212/300
4/4 [==============================] - 0s 20ms/step - loss: 0.3184 - accuracy:
0.9242 - val_loss: 1.1426 - val_accuracy: 0.5455
Epoch 213/300
4/4 [==============================] - 0s 20ms/step - loss: 0.3069 - accuracy:
0.9242 - val_loss: 1.0795 - val_accuracy: 0.5455
Epoch 214/300
4/4 [==============================] - 0s 20ms/step - loss: 0.3004 - accuracy:
0.9242 - val_loss: 1.0719 - val_accuracy: 0.5909
Epoch 215/300
4/4 [==============================] - 0s 14ms/step - loss: 0.2983 - accuracy:
0.9242 - val_loss: 1.0897 - val_accuracy: 0.5909
Epoch 216/300
4/4 [==============================] - 0s 19ms/step - loss: 0.2975 - accuracy:
0.9242 - val_loss: 1.0994 - val_accuracy: 0.5455
Epoch 217/300
4/4 [==============================] - 0s 19ms/step - loss: 0.2929 - accuracy:
0.9394 - val_loss: 1.0979 - val_accuracy: 0.5909
Epoch 218/300
4/4 [==============================] - 0s 18ms/step - loss: 0.2931 - accuracy:
0.9394 - val_loss: 1.1067 - val_accuracy: 0.5455
Epoch 219/300
4/4 [==============================] - 0s 19ms/step - loss: 0.2908 - accuracy:
0.9242 - val_loss: 1.1540 - val_accuracy: 0.5455
Epoch 220/300
4/4 [==============================] - 0s 20ms/step - loss: 0.2910 - accuracy:
0.9242 - val_loss: 1.1245 - val_accuracy: 0.5455
Epoch 221/300
4/4 [==============================] - 0s 19ms/step - loss: 0.2834 - accuracy:
0.9242 - val_loss: 1.1104 - val_accuracy: 0.5909
Epoch 222/300
4/4 [==============================] - 0s 21ms/step - loss: 0.2799 - accuracy:
0.9394 - val_loss: 1.0865 - val_accuracy: 0.5909
Epoch 223/300
4/4 [==============================] - 0s 18ms/step - loss: 0.2764 - accuracy:
0.9394 - val_loss: 1.1055 - val_accuracy: 0.5455
```

```
Epoch 224/300
4/4 [==============================] - 0s 20ms/step - loss: 0.2807 - accuracy:
0.9394 - val_loss: 1.0812 - val_accuracy: 0.6364
Epoch 225/300
4/4 [==============================] - 0s 20ms/step - loss: 0.2788 - accuracy:
0.9394 - val_loss: 1.0912 - val_accuracy: 0.6364
Epoch 226/300
4/4 [==============================] - 0s 19ms/step - loss: 0.2724 - accuracy:
0.9394 - val_loss: 1.0956 - val_accuracy: 0.6364
Epoch 227/300
4/4 [==============================] - 0s 15ms/step - loss: 0.2716 - accuracy:
0.9394 - val_loss: 1.1139 - val_accuracy: 0.5909
Epoch 228/300
4/4 [==============================] - 0s 15ms/step - loss: 0.2675 - accuracy:
0.9545 - val_loss: 1.0976 - val_accuracy: 0.6364
Epoch 229/300
4/4 [==============================] - 0s 14ms/step - loss: 0.2702 - accuracy:
0.9394 - val_loss: 1.0911 - val_accuracy: 0.6364
Epoch 230/300
4/4 [==============================] - 0s 21ms/step - loss: 0.2744 - accuracy:
0.9394 - val_loss: 1.0915 - val_accuracy: 0.5455
Epoch 231/300
4/4 [==============================] - 0s 17ms/step - loss: 0.2640 - accuracy:
0.9545 - val_loss: 1.1052 - val_accuracy: 0.5455
Epoch 232/300
4/4 [==============================] - 0s 15ms/step - loss: 0.2603 - accuracy:
0.9394 - val_loss: 1.0942 - val_accuracy: 0.5455
Epoch 233/300
4/4 [==============================] - 0s 19ms/step - loss: 0.2592 - accuracy:
0.9242 - val_loss: 1.0716 - val_accuracy: 0.6818
Epoch 234/300
4/4 [==============================] - 0s 19ms/step - loss: 0.2656 - accuracy:
0.9545 - val_loss: 1.0882 - val_accuracy: 0.6364
Epoch 235/300
4/4 [==============================] - 0s 21ms/step - loss: 0.2526 - accuracy:
0.9394 - val_loss: 1.1134 - val_accuracy: 0.5455
Epoch 236/300
4/4 [==============================] - 0s 13ms/step - loss: 0.2532 - accuracy:
0.9394 - val_loss: 1.1230 - val_accuracy: 0.5455
Epoch 237/300
4/4 [==============================] - 0s 20ms/step - loss: 0.2523 - accuracy:
0.9242 - val_loss: 1.1515 - val_accuracy: 0.5909
Epoch 238/300
4/4 [==============================] - 0s 18ms/step - loss: 0.2556 - accuracy:
0.9394 - val_loss: 1.1548 - val_accuracy: 0.5455
Epoch 239/300
4/4 [==============================] - 0s 15ms/step - loss: 0.2508 - accuracy:
0.9394 - val_loss: 1.0978 - val_accuracy: 0.6364
```

```
Epoch 240/300
4/4 [==============================] - 0s 21ms/step - loss: 0.2483 - accuracy:
0.9242 - val_loss: 1.1309 - val_accuracy: 0.5909
Epoch 241/300
4/4 [==============================] - 0s 16ms/step - loss: 0.2415 - accuracy:
0.9242 - val_loss: 1.1635 - val_accuracy: 0.5909
Epoch 242/300
4/4 [==============================] - 0s 15ms/step - loss: 0.2442 - accuracy:
0.9242 - val_loss: 1.1796 - val_accuracy: 0.5455
Epoch 243/300
4/4 [==============================] - 0s 16ms/step - loss: 0.2404 - accuracy:
0.9545 - val_loss: 1.1683 - val_accuracy: 0.5455
Epoch 244/300
4/4 [==============================] - 0s 21ms/step - loss: 0.2381 - accuracy:
0.9394 - val_loss: 1.1509 - val_accuracy: 0.5455
Epoch 245/300
4/4 [==============================] - 0s 15ms/step - loss: 0.2410 - accuracy:
0.9394 - val_loss: 1.1310 - val_accuracy: 0.5455
Epoch 246/300
4/4 [==============================] - 0s 15ms/step - loss: 0.2358 - accuracy:
0.9545 - val_loss: 1.1282 - val_accuracy: 0.5455
Epoch 247/300
4/4 [==============================] - 0s 14ms/step - loss: 0.2293 - accuracy:
0.9545 - val_loss: 1.1590 - val_accuracy: 0.5455
Epoch 248/300
4/4 [==============================] - 0s 19ms/step - loss: 0.2257 - accuracy:
0.9545 - val_loss: 1.1606 - val_accuracy: 0.5455
Epoch 249/300
4/4 [==============================] - 0s 16ms/step - loss: 0.2247 - accuracy:
0.9545 - val_loss: 1.1454 - val_accuracy: 0.6364
Epoch 250/300
4/4 [==============================] - 0s 15ms/step - loss: 0.2270 - accuracy:
0.9545 - val_loss: 1.1290 - val_accuracy: 0.6364
Epoch 251/300
4/4 [==============================] - 0s 15ms/step - loss: 0.2376 - accuracy:
0.9545 - val_loss: 1.1232 - val_accuracy: 0.6364
Epoch 252/300
4/4 [==============================] - 0s 22ms/step - loss: 0.2245 - accuracy:
0.9545 - val_loss: 1.1322 - val_accuracy: 0.6364
Epoch 253/300
4/4 [==============================] - 0s 20ms/step - loss: 0.2199 - accuracy:
0.9697 - val_loss: 1.1410 - val_accuracy: 0.6364
Epoch 254/300
4/4 [==============================] - 0s 16ms/step - loss: 0.2146 - accuracy:
0.9697 - val_loss: 1.1623 - val_accuracy: 0.5455
Epoch 255/300
4/4 [==============================] - 0s 21ms/step - loss: 0.2146 - accuracy:
0.9545 - val_loss: 1.1459 - val_accuracy: 0.6364
```

```
Epoch 256/300
4/4 [==============================] - 0s 14ms/step - loss: 0.2200 - accuracy:
0.9697 - val_loss: 1.1550 - val_accuracy: 0.6364
Epoch 257/300
4/4 [==============================] - 0s 21ms/step - loss: 0.2090 - accuracy:
0.9697 - val_loss: 1.1811 - val_accuracy: 0.5455
Epoch 258/300
4/4 [==============================] - 0s 14ms/step - loss: 0.2109 - accuracy:
0.9697 - val_loss: 1.1988 - val_accuracy: 0.5455
Epoch 259/300
4/4 [==============================] - 0s 19ms/step - loss: 0.2077 - accuracy:
0.9545 - val_loss: 1.1915 - val_accuracy: 0.5455
Epoch 260/300
4/4 [==============================] - 0s 20ms/step - loss: 0.2048 - accuracy:
0.9545 - val_loss: 1.2031 - val_accuracy: 0.5909
Epoch 261/300
4/4 [==============================] - 0s 14ms/step - loss: 0.2058 - accuracy:
0.9545 - val_loss: 1.1661 - val_accuracy: 0.6364
Epoch 262/300
4/4 [==============================] - 0s 19ms/step - loss: 0.2019 - accuracy:
0.9697 - val_loss: 1.1780 - val_accuracy: 0.5909
Epoch 263/300
4/4 [==============================] - 0s 14ms/step - loss: 0.2051 - accuracy:
0.9697 - val_loss: 1.2044 - val_accuracy: 0.6364
Epoch 264/300
4/4 [==============================] - 0s 21ms/step - loss: 0.1989 - accuracy:
0.9697 - val_loss: 1.1913 - val_accuracy: 0.6364
Epoch 265/300
4/4 [==============================] - 0s 20ms/step - loss: 0.1993 - accuracy:
0.9697 - val_loss: 1.2031 - val_accuracy: 0.6364
Epoch 266/300
4/4 [==============================] - 0s 15ms/step - loss: 0.1942 - accuracy:
0.9697 - val_loss: 1.1950 - val_accuracy: 0.5455
Epoch 267/300
4/4 [==============================] - 0s 24ms/step - loss: 0.1928 - accuracy:
0.9697 - val_loss: 1.1948 - val_accuracy: 0.5909
Epoch 268/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1946 - accuracy:
0.9848 - val_loss: 1.2106 - val_accuracy: 0.5455
Epoch 269/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1913 - accuracy:
0.9697 - val_loss: 1.1841 - val_accuracy: 0.6364
Epoch 270/300
4/4 [==============================] - 0s 15ms/step - loss: 0.1986 - accuracy:
0.9545 - val_loss: 1.1704 - val_accuracy: 0.6364
Epoch 271/300
4/4 [==============================] - 0s 18ms/step - loss: 0.1912 - accuracy:
0.9545 - val_loss: 1.1701 - val_accuracy: 0.6364
```

```
Epoch 272/300
4/4 [==============================] - 0s 17ms/step - loss: 0.1995 - accuracy:
0.9697 - val_loss: 1.2162 - val_accuracy: 0.5455
Epoch 273/300
4/4 [==============================] - 0s 16ms/step - loss: 0.1868 - accuracy:
0.9848 - val_loss: 1.2130 - val_accuracy: 0.5455
Epoch 274/300
4/4 [==============================] - 0s 21ms/step - loss: 0.1850 - accuracy:
0.9697 - val_loss: 1.1971 - val_accuracy: 0.6364
Epoch 275/300
4/4 [==============================] - 0s 16ms/step - loss: 0.1900 - accuracy:
0.9697 - val_loss: 1.2179 - val_accuracy: 0.6818
Epoch 276/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1840 - accuracy:
0.9848 - val_loss: 1.1976 - val_accuracy: 0.6364
Epoch 277/300
4/4 [==============================] - 0s 16ms/step - loss: 0.1791 - accuracy:
0.9697 - val_loss: 1.1920 - val_accuracy: 0.6364
Epoch 278/300
4/4 [==============================] - 0s 17ms/step - loss: 0.1784 - accuracy:
0.9697 - val_loss: 1.2141 - val_accuracy: 0.5455
Epoch 279/300
4/4 [==============================] - 0s 19ms/step - loss: 0.1759 - accuracy:
0.9848 - val_loss: 1.2216 - val_accuracy: 0.5455
Epoch 280/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1759 - accuracy:
0.9697 - val_loss: 1.2307 - val_accuracy: 0.5455
Epoch 281/300
4/4 [==============================] - 0s 20ms/step - loss: 0.1729 - accuracy:
0.9848 - val_loss: 1.2304 - val_accuracy: 0.5455
Epoch 282/300
4/4 [==============================] - 0s 18ms/step - loss: 0.1705 - accuracy:
0.9848 - val_loss: 1.2367 - val_accuracy: 0.5909
Epoch 283/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1715 - accuracy:
0.9848 - val_loss: 1.2550 - val_accuracy: 0.5909
Epoch 284/300
4/4 [==============================] - 0s 21ms/step - loss: 0.1732 - accuracy:
0.9848 - val_loss: 1.2746 - val_accuracy: 0.5909
Epoch 285/300
4/4 [==============================] - 0s 20ms/step - loss: 0.1718 - accuracy:
0.9848 - val_loss: 1.2565 - val_accuracy: 0.6818
Epoch 286/300
4/4 [==============================] - 0s 16ms/step - loss: 0.1676 - accuracy:
0.9848 - val_loss: 1.2392 - val_accuracy: 0.6364
Epoch 287/300
4/4 [==============================] - 0s 24ms/step - loss: 0.1632 - accuracy:
0.9697 - val_loss: 1.2592 - val_accuracy: 0.5909
```

```
Epoch 288/300
4/4 [==============================] - 0s 16ms/step - loss: 0.1647 - accuracy:
0.9848 - val_loss: 1.2366 - val_accuracy: 0.6364
Epoch 289/300
4/4 [==============================] - 0s 15ms/step - loss: 0.1640 - accuracy:
0.9545 - val_loss: 1.2702 - val_accuracy: 0.5909
Epoch 290/300
4/4 [==============================] - 0s 16ms/step - loss: 0.1627 - accuracy:
0.9848 - val_loss: 1.2568 - val_accuracy: 0.6364
Epoch 291/300
4/4 [==============================] - 0s 20ms/step - loss: 0.1594 - accuracy:
0.9848 - val_loss: 1.2283 - val_accuracy: 0.6364
Epoch 292/300
4/4 [==============================] - 0s 21ms/step - loss: 0.1615 - accuracy:
0.9697 - val_loss: 1.2657 - val_accuracy: 0.5909
Epoch 293/300
4/4 [==============================] - 0s 15ms/step - loss: 0.1573 - accuracy:
0.9848 - val_loss: 1.2575 - val_accuracy: 0.5909
Epoch 294/300
4/4 [==============================] - 0s 15ms/step - loss: 0.1563 - accuracy:
0.9848 - val_loss: 1.2501 - val_accuracy: 0.5909
Epoch 295/300
4/4 [==============================] - 0s 15ms/step - loss: 0.1544 - accuracy:
0.9848 - val_loss: 1.2639 - val_accuracy: 0.6364
Epoch 296/300
4/4 [==============================] - 0s 20ms/step - loss: 0.1535 - accuracy:
0.9848 - val_loss: 1.2747 - val_accuracy: 0.5455
Epoch 297/300
4/4 [==============================] - 0s 20ms/step - loss: 0.1529 - accuracy:
0.9848 - val_loss: 1.2648 - val_accuracy: 0.5909
Epoch 298/300
4/4 [==============================] - 0s 20ms/step - loss: 0.1533 - accuracy:
0.9848 - val_loss: 1.2488 - val_accuracy: 0.6364
Epoch 299/300
4/4 [==============================] - 0s 17ms/step - loss: 0.1575 - accuracy:
0.9848 - val_loss: 1.2662 - val_accuracy: 0.6364
Epoch 300/300
4/4 [==============================] - 0s 20ms/step - loss: 0.1495 - accuracy:
0.9697 - val_loss: 1.2502 - val_accuracy: 0.6364
```

```
[72]: <keras.src.callbacks.History at 0x7bf124acb1c0>
```

Which activations did you use in the middle layers? Why was softmax used in the last layer?

```
[73]: ## Your answer: I use the Relu in the middle layer. Softmax activation is used␣
      ↪in the last layer to handle multi-class classification, because the function␣
      ↪has range over [0,5], unlike sigmoid function only range over +-1.
```

Was it a good idea to train for 300 epochs? Should you train a bit more? Why or why not?

```
[74]: ## Your answer: For our data specially, we can train over 300 epochs because we␣
      ↪have a small training set because small batch size usually in better␣
      ↪optimization. But for large dataset, we need to have fewer epochs to␣
      ↪minimize optimization time.
```

Why is loss='categorical_crossentropy' and optimizer='sgd'? Would you want to change something? Why / Why not?

```
[75]: ## Your answer:It says that we are using crossentropy as our loss function and␣
      ↪stochastic gradient descent as our optimizing method. We can change the␣
      ↪optimizer to be gradient descent sometimes. But we ought to use␣
      ↪cross-entrophy function since the function is always convex (has just 1␣
      ↪extrema). Other loss functions may  not be convex
```

Can you try getting the model's training history out and plotting the curves?

```
[76]: ## Your code to plot training and validation curves in a single plot (Make␣
      ↪changes in the model cell to be able to do this)
      history = keras_model.fit(preprocessor(X_train), y_train, batch_size=20,␣
      ↪epochs=300, validation_split=0.25)

      # Plot training and validation curves
      plt.figure(figsize=(12, 6))
      plt.plot(history.history['accuracy'], label='Training Accuracy')
      plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
      plt.plot(history.history['loss'], label='Training Loss')
      plt.plot(history.history['val_loss'], label='Validation Loss')
      plt.xlabel('Epoch')
      plt.ylabel('Accuracy / Loss')
      plt.title('Training and Validation Curves')
      plt.legend()
      plt.show()
```

```
Epoch 1/300
4/4 [==============================] - 0s 33ms/step - loss: 0.1487 - accuracy:
0.9697 - val_loss: 1.3090 - val_accuracy: 0.5455
Epoch 2/300
4/4 [==============================] - 0s 18ms/step - loss: 0.1497 - accuracy:
0.9848 - val_loss: 1.2858 - val_accuracy: 0.5455
Epoch 3/300
4/4 [==============================] - 0s 22ms/step - loss: 0.1467 - accuracy:
0.9848 - val_loss: 1.2720 - val_accuracy: 0.6364
Epoch 4/300
4/4 [==============================] - 0s 19ms/step - loss: 0.1480 - accuracy:
0.9848 - val_loss: 1.2745 - val_accuracy: 0.6364
Epoch 5/300
```

```
4/4 [==============================] - 0s 19ms/step - loss: 0.1431 - accuracy:
0.9848 - val_loss: 1.2847 - val_accuracy: 0.6364
Epoch 6/300
4/4 [==============================] - 0s 15ms/step - loss: 0.1473 - accuracy:
0.9848 - val_loss: 1.2468 - val_accuracy: 0.6364
Epoch 7/300
4/4 [==============================] - 0s 19ms/step - loss: 0.1535 - accuracy:
0.9697 - val_loss: 1.2731 - val_accuracy: 0.6364
Epoch 8/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1406 - accuracy:
0.9697 - val_loss: 1.2690 - val_accuracy: 0.6364
Epoch 9/300
4/4 [==============================] - 0s 19ms/step - loss: 0.1370 - accuracy:
0.9697 - val_loss: 1.2842 - val_accuracy: 0.6364
Epoch 10/300
4/4 [==============================] - 0s 13ms/step - loss: 0.1361 - accuracy:
0.9697 - val_loss: 1.3004 - val_accuracy: 0.5909
Epoch 11/300
4/4 [==============================] - 0s 19ms/step - loss: 0.1352 - accuracy:
0.9848 - val_loss: 1.2770 - val_accuracy: 0.6364
Epoch 12/300
4/4 [==============================] - 0s 19ms/step - loss: 0.1371 - accuracy:
0.9697 - val_loss: 1.2874 - val_accuracy: 0.6364
Epoch 13/300
4/4 [==============================] - 0s 13ms/step - loss: 0.1322 - accuracy:
0.9697 - val_loss: 1.3191 - val_accuracy: 0.5455
Epoch 14/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1321 - accuracy:
0.9848 - val_loss: 1.3048 - val_accuracy: 0.6818
Epoch 15/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1324 - accuracy:
0.9848 - val_loss: 1.3240 - val_accuracy: 0.6818
Epoch 16/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1323 - accuracy:
0.9848 - val_loss: 1.2903 - val_accuracy: 0.6364
Epoch 17/300
4/4 [==============================] - 0s 13ms/step - loss: 0.1360 - accuracy:
0.9848 - val_loss: 1.3115 - val_accuracy: 0.5455
Epoch 18/300
4/4 [==============================] - 0s 15ms/step - loss: 0.1290 - accuracy:
0.9848 - val_loss: 1.3498 - val_accuracy: 0.5455
Epoch 19/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1310 - accuracy:
0.9848 - val_loss: 1.3668 - val_accuracy: 0.5455
Epoch 20/300
4/4 [==============================] - 0s 15ms/step - loss: 0.1309 - accuracy:
0.9848 - val_loss: 1.3608 - val_accuracy: 0.5455
Epoch 21/300
```

```
4/4 [==============================] - 0s 14ms/step - loss: 0.1242 - accuracy:
0.9848 - val_loss: 1.3566 - val_accuracy: 0.6364
Epoch 22/300
4/4 [==============================] - 0s 13ms/step - loss: 0.1259 - accuracy:
0.9697 - val_loss: 1.3629 - val_accuracy: 0.5455
Epoch 23/300
4/4 [==============================] - 0s 13ms/step - loss: 0.1254 - accuracy:
0.9697 - val_loss: 1.3666 - val_accuracy: 0.5455
Epoch 24/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1217 - accuracy:
0.9848 - val_loss: 1.3406 - val_accuracy: 0.5455
Epoch 25/300
4/4 [==============================] - 0s 13ms/step - loss: 0.1201 - accuracy:
0.9848 - val_loss: 1.3355 - val_accuracy: 0.6364
Epoch 26/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1181 - accuracy:
0.9848 - val_loss: 1.3603 - val_accuracy: 0.5909
Epoch 27/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1222 - accuracy:
0.9697 - val_loss: 1.3711 - val_accuracy: 0.5909
Epoch 28/300
4/4 [==============================] - 0s 19ms/step - loss: 0.1189 - accuracy:
0.9848 - val_loss: 1.3705 - val_accuracy: 0.6364
Epoch 29/300
4/4 [==============================] - 0s 20ms/step - loss: 0.1170 - accuracy:
1.0000 - val_loss: 1.3709 - val_accuracy: 0.6364
Epoch 30/300
4/4 [==============================] - 0s 19ms/step - loss: 0.1168 - accuracy:
0.9848 - val_loss: 1.3518 - val_accuracy: 0.6364
Epoch 31/300
4/4 [==============================] - 0s 13ms/step - loss: 0.1223 - accuracy:
0.9848 - val_loss: 1.3735 - val_accuracy: 0.6364
Epoch 32/300
4/4 [==============================] - 0s 18ms/step - loss: 0.1184 - accuracy:
0.9848 - val_loss: 1.3684 - val_accuracy: 0.6364
Epoch 33/300
4/4 [==============================] - 0s 15ms/step - loss: 0.1165 - accuracy:
0.9848 - val_loss: 1.3517 - val_accuracy: 0.6364
Epoch 34/300
4/4 [==============================] - 0s 18ms/step - loss: 0.1154 - accuracy:
0.9848 - val_loss: 1.4190 - val_accuracy: 0.5000
Epoch 35/300
4/4 [==============================] - 0s 18ms/step - loss: 0.1166 - accuracy:
0.9848 - val_loss: 1.3688 - val_accuracy: 0.6364
Epoch 36/300
4/4 [==============================] - 0s 13ms/step - loss: 0.1158 - accuracy:
0.9848 - val_loss: 1.4054 - val_accuracy: 0.5455
Epoch 37/300
```

```
4/4 [==============================] - 0s 14ms/step - loss: 0.1129 - accuracy:
0.9848 - val_loss: 1.4440 - val_accuracy: 0.5455
Epoch 38/300
4/4 [==============================] - 0s 20ms/step - loss: 0.1149 - accuracy:
0.9848 - val_loss: 1.3891 - val_accuracy: 0.5455
Epoch 39/300
4/4 [==============================] - 0s 18ms/step - loss: 0.1099 - accuracy:
0.9848 - val_loss: 1.4000 - val_accuracy: 0.5455
Epoch 40/300
4/4 [==============================] - 0s 18ms/step - loss: 0.1079 - accuracy:
0.9848 - val_loss: 1.4087 - val_accuracy: 0.6364
Epoch 41/300
4/4 [==============================] - 0s 18ms/step - loss: 0.1064 - accuracy:
1.0000 - val_loss: 1.4113 - val_accuracy: 0.6364
Epoch 42/300
4/4 [==============================] - 0s 18ms/step - loss: 0.1073 - accuracy:
0.9848 - val_loss: 1.3971 - val_accuracy: 0.6364
Epoch 43/300
4/4 [==============================] - 0s 18ms/step - loss: 0.1049 - accuracy:
0.9848 - val_loss: 1.3979 - val_accuracy: 0.6364
Epoch 44/300
4/4 [==============================] - 0s 19ms/step - loss: 0.1058 - accuracy:
0.9848 - val_loss: 1.4117 - val_accuracy: 0.6364
Epoch 45/300
4/4 [==============================] - 0s 20ms/step - loss: 0.1030 - accuracy:
1.0000 - val_loss: 1.4369 - val_accuracy: 0.5455
Epoch 46/300
4/4 [==============================] - 0s 12ms/step - loss: 0.1052 - accuracy:
0.9848 - val_loss: 1.4169 - val_accuracy: 0.6364
Epoch 47/300
4/4 [==============================] - 0s 15ms/step - loss: 0.1014 - accuracy:
1.0000 - val_loss: 1.4028 - val_accuracy: 0.6364
Epoch 48/300
4/4 [==============================] - 0s 12ms/step - loss: 0.1027 - accuracy:
0.9848 - val_loss: 1.4258 - val_accuracy: 0.6364
Epoch 49/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1013 - accuracy:
0.9848 - val_loss: 1.4013 - val_accuracy: 0.6364
Epoch 50/300
4/4 [==============================] - 0s 14ms/step - loss: 0.1068 - accuracy:
0.9848 - val_loss: 1.4115 - val_accuracy: 0.6364
Epoch 51/300
4/4 [==============================] - 0s 19ms/step - loss: 0.1070 - accuracy:
0.9848 - val_loss: 1.4036 - val_accuracy: 0.6364
Epoch 52/300
4/4 [==============================] - 0s 13ms/step - loss: 0.1019 - accuracy:
0.9848 - val_loss: 1.4165 - val_accuracy: 0.6364
Epoch 53/300
```

```
4/4 [==============================] - 0s 14ms/step - loss: 0.1024 - accuracy:
0.9848 - val_loss: 1.4136 - val_accuracy: 0.6364
Epoch 54/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0972 - accuracy:
0.9848 - val_loss: 1.4288 - val_accuracy: 0.6364
Epoch 55/300
4/4 [==============================] - 0s 17ms/step - loss: 0.0981 - accuracy:
0.9848 - val_loss: 1.3995 - val_accuracy: 0.6364
Epoch 56/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0978 - accuracy:
0.9848 - val_loss: 1.4268 - val_accuracy: 0.5000
Epoch 57/300
4/4 [==============================] - 0s 13ms/step - loss: 0.0960 - accuracy:
0.9848 - val_loss: 1.4262 - val_accuracy: 0.5455
Epoch 58/300
4/4 [==============================] - 0s 13ms/step - loss: 0.0943 - accuracy:
0.9848 - val_loss: 1.4179 - val_accuracy: 0.6364
Epoch 59/300
4/4 [==============================] - 0s 13ms/step - loss: 0.0939 - accuracy:
1.0000 - val_loss: 1.4324 - val_accuracy: 0.6364
Epoch 60/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0949 - accuracy:
0.9848 - val_loss: 1.4357 - val_accuracy: 0.6364
Epoch 61/300
4/4 [==============================] - 0s 13ms/step - loss: 0.0938 - accuracy:
0.9848 - val_loss: 1.4819 - val_accuracy: 0.5000
Epoch 62/300
4/4 [==============================] - 0s 29ms/step - loss: 0.0977 - accuracy:
0.9848 - val_loss: 1.4708 - val_accuracy: 0.5455
Epoch 63/300
4/4 [==============================] - 0s 34ms/step - loss: 0.0917 - accuracy:
0.9848 - val_loss: 1.4716 - val_accuracy: 0.5455
Epoch 64/300
4/4 [==============================] - 0s 30ms/step - loss: 0.0933 - accuracy:
1.0000 - val_loss: 1.4746 - val_accuracy: 0.5455
Epoch 65/300
4/4 [==============================] - 0s 41ms/step - loss: 0.0918 - accuracy:
0.9848 - val_loss: 1.4730 - val_accuracy: 0.5909
Epoch 66/300
4/4 [==============================] - 0s 27ms/step - loss: 0.0884 - accuracy:
1.0000 - val_loss: 1.4681 - val_accuracy: 0.6364
Epoch 67/300
4/4 [==============================] - 0s 33ms/step - loss: 0.0894 - accuracy:
0.9848 - val_loss: 1.4677 - val_accuracy: 0.5909
Epoch 68/300
4/4 [==============================] - 0s 41ms/step - loss: 0.0902 - accuracy:
0.9848 - val_loss: 1.4547 - val_accuracy: 0.6364
Epoch 69/300
```

```
4/4 [==============================] - 0s 29ms/step - loss: 0.0876 - accuracy:
0.9848 - val_loss: 1.4575 - val_accuracy: 0.6364
Epoch 70/300
4/4 [==============================] - 0s 30ms/step - loss: 0.0860 - accuracy:
1.0000 - val_loss: 1.4891 - val_accuracy: 0.5455
Epoch 71/300
4/4 [==============================] - 0s 29ms/step - loss: 0.0892 - accuracy:
1.0000 - val_loss: 1.5011 - val_accuracy: 0.5455
Epoch 72/300
4/4 [==============================] - 0s 33ms/step - loss: 0.0856 - accuracy:
1.0000 - val_loss: 1.4575 - val_accuracy: 0.6364
Epoch 73/300
4/4 [==============================] - 0s 34ms/step - loss: 0.0859 - accuracy:
1.0000 - val_loss: 1.4327 - val_accuracy: 0.6364
Epoch 74/300
4/4 [==============================] - 0s 24ms/step - loss: 0.0903 - accuracy:
0.9848 - val_loss: 1.4529 - val_accuracy: 0.6364
Epoch 75/300
4/4 [==============================] - 0s 37ms/step - loss: 0.0838 - accuracy:
0.9848 - val_loss: 1.4562 - val_accuracy: 0.6364
Epoch 76/300
4/4 [==============================] - 0s 57ms/step - loss: 0.0842 - accuracy:
0.9848 - val_loss: 1.4776 - val_accuracy: 0.6364
Epoch 77/300
4/4 [==============================] - 0s 33ms/step - loss: 0.0840 - accuracy:
1.0000 - val_loss: 1.4539 - val_accuracy: 0.6364
Epoch 78/300
4/4 [==============================] - 0s 40ms/step - loss: 0.0898 - accuracy:
0.9848 - val_loss: 1.4618 - val_accuracy: 0.6364
Epoch 79/300
4/4 [==============================] - 0s 29ms/step - loss: 0.0847 - accuracy:
0.9848 - val_loss: 1.4670 - val_accuracy: 0.6364
Epoch 80/300
4/4 [==============================] - 0s 52ms/step - loss: 0.0825 - accuracy:
0.9848 - val_loss: 1.4533 - val_accuracy: 0.6364
Epoch 81/300
4/4 [==============================] - 0s 26ms/step - loss: 0.0802 - accuracy:
1.0000 - val_loss: 1.4646 - val_accuracy: 0.6364
Epoch 82/300
4/4 [==============================] - 0s 27ms/step - loss: 0.0818 - accuracy:
0.9848 - val_loss: 1.4558 - val_accuracy: 0.6364
Epoch 83/300
4/4 [==============================] - 0s 33ms/step - loss: 0.0843 - accuracy:
0.9848 - val_loss: 1.4679 - val_accuracy: 0.6364
Epoch 84/300
4/4 [==============================] - 0s 47ms/step - loss: 0.0865 - accuracy:
0.9848 - val_loss: 1.4694 - val_accuracy: 0.6364
Epoch 85/300
```

```
4/4 [==============================] - 0s 71ms/step - loss: 0.0812 - accuracy:
0.9848 - val_loss: 1.4796 - val_accuracy: 0.6364
Epoch 86/300
4/4 [==============================] - 0s 64ms/step - loss: 0.0775 - accuracy:
1.0000 - val_loss: 1.4870 - val_accuracy: 0.6364
Epoch 87/300
4/4 [==============================] - 0s 69ms/step - loss: 0.0791 - accuracy:
0.9848 - val_loss: 1.4986 - val_accuracy: 0.5909
Epoch 88/300
4/4 [==============================] - 0s 73ms/step - loss: 0.0770 - accuracy:
0.9848 - val_loss: 1.5249 - val_accuracy: 0.5000
Epoch 89/300
4/4 [==============================] - 0s 73ms/step - loss: 0.0805 - accuracy:
0.9848 - val_loss: 1.5121 - val_accuracy: 0.5000
Epoch 90/300
4/4 [==============================] - 0s 79ms/step - loss: 0.0779 - accuracy:
0.9848 - val_loss: 1.5053 - val_accuracy: 0.5000
Epoch 91/300
4/4 [==============================] - 0s 84ms/step - loss: 0.0762 - accuracy:
1.0000 - val_loss: 1.5371 - val_accuracy: 0.5000
Epoch 92/300
4/4 [==============================] - 0s 64ms/step - loss: 0.0782 - accuracy:
0.9848 - val_loss: 1.5036 - val_accuracy: 0.5455
Epoch 93/300
4/4 [==============================] - 0s 70ms/step - loss: 0.0742 - accuracy:
1.0000 - val_loss: 1.5157 - val_accuracy: 0.5909
Epoch 94/300
4/4 [==============================] - 0s 72ms/step - loss: 0.0741 - accuracy:
1.0000 - val_loss: 1.5520 - val_accuracy: 0.5000
Epoch 95/300
4/4 [==============================] - 0s 77ms/step - loss: 0.0782 - accuracy:
0.9848 - val_loss: 1.5319 - val_accuracy: 0.5000
Epoch 96/300
4/4 [==============================] - 0s 70ms/step - loss: 0.0749 - accuracy:
0.9848 - val_loss: 1.5329 - val_accuracy: 0.5455
Epoch 97/300
4/4 [==============================] - 0s 59ms/step - loss: 0.0735 - accuracy:
0.9848 - val_loss: 1.5340 - val_accuracy: 0.5455
Epoch 98/300
4/4 [==============================] - 0s 47ms/step - loss: 0.0712 - accuracy:
1.0000 - val_loss: 1.5303 - val_accuracy: 0.5909
Epoch 99/300
4/4 [==============================] - 0s 83ms/step - loss: 0.0712 - accuracy:
1.0000 - val_loss: 1.5410 - val_accuracy: 0.5909
Epoch 100/300
4/4 [==============================] - 0s 79ms/step - loss: 0.0722 - accuracy:
1.0000 - val_loss: 1.5238 - val_accuracy: 0.6364
Epoch 101/300
```

```
4/4 [==============================] - 0s 59ms/step - loss: 0.0690 - accuracy:
1.0000 - val_loss: 1.5287 - val_accuracy: 0.6364
Epoch 102/300
4/4 [==============================] - 0s 67ms/step - loss: 0.0710 - accuracy:
1.0000 - val_loss: 1.5215 - val_accuracy: 0.6364
Epoch 103/300
4/4 [==============================] - 0s 51ms/step - loss: 0.0683 - accuracy:
1.0000 - val_loss: 1.5323 - val_accuracy: 0.6364
Epoch 104/300
4/4 [==============================] - 0s 45ms/step - loss: 0.0706 - accuracy:
1.0000 - val_loss: 1.5086 - val_accuracy: 0.6364
Epoch 105/300
4/4 [==============================] - 0s 73ms/step - loss: 0.0765 - accuracy:
0.9848 - val_loss: 1.5175 - val_accuracy: 0.6364
Epoch 106/300
4/4 [==============================] - 0s 31ms/step - loss: 0.0695 - accuracy:
0.9848 - val_loss: 1.5204 - val_accuracy: 0.6364
Epoch 107/300
4/4 [==============================] - 0s 27ms/step - loss: 0.0685 - accuracy:
0.9848 - val_loss: 1.5492 - val_accuracy: 0.5000
Epoch 108/300
4/4 [==============================] - 0s 44ms/step - loss: 0.0678 - accuracy:
0.9848 - val_loss: 1.5512 - val_accuracy: 0.5909
Epoch 109/300
4/4 [==============================] - 0s 53ms/step - loss: 0.0667 - accuracy:
1.0000 - val_loss: 1.5443 - val_accuracy: 0.6364
Epoch 110/300
4/4 [==============================] - 0s 49ms/step - loss: 0.0679 - accuracy:
1.0000 - val_loss: 1.5641 - val_accuracy: 0.5909
Epoch 111/300
4/4 [==============================] - 0s 56ms/step - loss: 0.0668 - accuracy:
1.0000 - val_loss: 1.5765 - val_accuracy: 0.5909
Epoch 112/300
4/4 [==============================] - 0s 50ms/step - loss: 0.0662 - accuracy:
1.0000 - val_loss: 1.5732 - val_accuracy: 0.5909
Epoch 113/300
4/4 [==============================] - 0s 41ms/step - loss: 0.0641 - accuracy:
1.0000 - val_loss: 1.5551 - val_accuracy: 0.5909
Epoch 114/300
4/4 [==============================] - 0s 61ms/step - loss: 0.0637 - accuracy:
1.0000 - val_loss: 1.5716 - val_accuracy: 0.5455
Epoch 115/300
4/4 [==============================] - 0s 41ms/step - loss: 0.0645 - accuracy:
1.0000 - val_loss: 1.5594 - val_accuracy: 0.6364
Epoch 116/300
4/4 [==============================] - 0s 28ms/step - loss: 0.0629 - accuracy:
1.0000 - val_loss: 1.5682 - val_accuracy: 0.5909
Epoch 117/300
```

```
4/4 [==============================] - 0s 44ms/step - loss: 0.0642 - accuracy:
1.0000 - val_loss: 1.5616 - val_accuracy: 0.5455
Epoch 118/300
4/4 [==============================] - 0s 38ms/step - loss: 0.0635 - accuracy:
1.0000 - val_loss: 1.5969 - val_accuracy: 0.5000
Epoch 119/300
4/4 [==============================] - 0s 60ms/step - loss: 0.0648 - accuracy:
0.9848 - val_loss: 1.5750 - val_accuracy: 0.5455
Epoch 120/300
4/4 [==============================] - 0s 57ms/step - loss: 0.0636 - accuracy:
1.0000 - val_loss: 1.5712 - val_accuracy: 0.5909
Epoch 121/300
4/4 [==============================] - 0s 42ms/step - loss: 0.0633 - accuracy:
1.0000 - val_loss: 1.5909 - val_accuracy: 0.5909
Epoch 122/300
4/4 [==============================] - 0s 38ms/step - loss: 0.0637 - accuracy:
1.0000 - val_loss: 1.6109 - val_accuracy: 0.5455
Epoch 123/300
4/4 [==============================] - 0s 28ms/step - loss: 0.0626 - accuracy:
1.0000 - val_loss: 1.5939 - val_accuracy: 0.5909
Epoch 124/300
4/4 [==============================] - 0s 27ms/step - loss: 0.0618 - accuracy:
1.0000 - val_loss: 1.5673 - val_accuracy: 0.5909
Epoch 125/300
4/4 [==============================] - 0s 41ms/step - loss: 0.0605 - accuracy:
1.0000 - val_loss: 1.5696 - val_accuracy: 0.6364
Epoch 126/300
4/4 [==============================] - 0s 53ms/step - loss: 0.0610 - accuracy:
1.0000 - val_loss: 1.5556 - val_accuracy: 0.6364
Epoch 127/300
4/4 [==============================] - 0s 54ms/step - loss: 0.0605 - accuracy:
1.0000 - val_loss: 1.6096 - val_accuracy: 0.5000
Epoch 128/300
4/4 [==============================] - 0s 48ms/step - loss: 0.0623 - accuracy:
0.9848 - val_loss: 1.5879 - val_accuracy: 0.5909
Epoch 129/300
4/4 [==============================] - 0s 30ms/step - loss: 0.0600 - accuracy:
1.0000 - val_loss: 1.6178 - val_accuracy: 0.5455
Epoch 130/300
4/4 [==============================] - 0s 48ms/step - loss: 0.0583 - accuracy:
1.0000 - val_loss: 1.5831 - val_accuracy: 0.6364
Epoch 131/300
4/4 [==============================] - 0s 45ms/step - loss: 0.0588 - accuracy:
1.0000 - val_loss: 1.6193 - val_accuracy: 0.5000
Epoch 132/300
4/4 [==============================] - 0s 57ms/step - loss: 0.0605 - accuracy:
0.9848 - val_loss: 1.5877 - val_accuracy: 0.6364
Epoch 133/300
```

```
4/4 [==============================] - 0s 58ms/step - loss: 0.0593 - accuracy:
0.9848 - val_loss: 1.5931 - val_accuracy: 0.6364
Epoch 134/300
4/4 [==============================] - 0s 31ms/step - loss: 0.0579 - accuracy:
1.0000 - val_loss: 1.6133 - val_accuracy: 0.5909
Epoch 135/300
4/4 [==============================] - 0s 33ms/step - loss: 0.0554 - accuracy:
1.0000 - val_loss: 1.5961 - val_accuracy: 0.5909
Epoch 136/300
4/4 [==============================] - 0s 36ms/step - loss: 0.0576 - accuracy:
1.0000 - val_loss: 1.6237 - val_accuracy: 0.5455
Epoch 137/300
4/4 [==============================] - 0s 30ms/step - loss: 0.0565 - accuracy:
1.0000 - val_loss: 1.6319 - val_accuracy: 0.5455
Epoch 138/300
4/4 [==============================] - 0s 29ms/step - loss: 0.0572 - accuracy:
1.0000 - val_loss: 1.6127 - val_accuracy: 0.5909
Epoch 139/300
4/4 [==============================] - 0s 27ms/step - loss: 0.0555 - accuracy:
1.0000 - val_loss: 1.6184 - val_accuracy: 0.5909
Epoch 140/300
4/4 [==============================] - 0s 58ms/step - loss: 0.0566 - accuracy:
1.0000 - val_loss: 1.6016 - val_accuracy: 0.6364
Epoch 141/300
4/4 [==============================] - 0s 33ms/step - loss: 0.0577 - accuracy:
0.9848 - val_loss: 1.6191 - val_accuracy: 0.6364
Epoch 142/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0551 - accuracy:
0.9848 - val_loss: 1.6640 - val_accuracy: 0.5000
Epoch 143/300
4/4 [==============================] - 0s 23ms/step - loss: 0.0547 - accuracy:
1.0000 - val_loss: 1.6351 - val_accuracy: 0.5455
Epoch 144/300
4/4 [==============================] - 0s 17ms/step - loss: 0.0545 - accuracy:
1.0000 - val_loss: 1.6348 - val_accuracy: 0.5909
Epoch 145/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0532 - accuracy:
1.0000 - val_loss: 1.6205 - val_accuracy: 0.5909
Epoch 146/300
4/4 [==============================] - 0s 18ms/step - loss: 0.0538 - accuracy:
1.0000 - val_loss: 1.6276 - val_accuracy: 0.5909
Epoch 147/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0514 - accuracy:
1.0000 - val_loss: 1.6295 - val_accuracy: 0.5909
Epoch 148/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0512 - accuracy:
1.0000 - val_loss: 1.6379 - val_accuracy: 0.5455
Epoch 149/300
```

```
4/4 [==============================] - 0s 21ms/step - loss: 0.0526 - accuracy:
1.0000 - val_loss: 1.6081 - val_accuracy: 0.6364
Epoch 150/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0548 - accuracy:
0.9848 - val_loss: 1.6470 - val_accuracy: 0.5000
Epoch 151/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0513 - accuracy:
1.0000 - val_loss: 1.6361 - val_accuracy: 0.5455
Epoch 152/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0503 - accuracy:
1.0000 - val_loss: 1.6393 - val_accuracy: 0.5455
Epoch 153/300
4/4 [==============================] - 0s 16ms/step - loss: 0.0497 - accuracy:
1.0000 - val_loss: 1.6371 - val_accuracy: 0.5909
Epoch 154/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0507 - accuracy:
1.0000 - val_loss: 1.6259 - val_accuracy: 0.5909
Epoch 155/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0501 - accuracy:
1.0000 - val_loss: 1.6518 - val_accuracy: 0.5455
Epoch 156/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0505 - accuracy:
1.0000 - val_loss: 1.6542 - val_accuracy: 0.5455
Epoch 157/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0495 - accuracy:
1.0000 - val_loss: 1.6375 - val_accuracy: 0.6364
Epoch 158/300
4/4 [==============================] - 0s 21ms/step - loss: 0.0510 - accuracy:
0.9848 - val_loss: 1.6477 - val_accuracy: 0.5909
Epoch 159/300
4/4 [==============================] - 0s 23ms/step - loss: 0.0482 - accuracy:
1.0000 - val_loss: 1.6471 - val_accuracy: 0.5909
Epoch 160/300
4/4 [==============================] - 0s 23ms/step - loss: 0.0490 - accuracy:
1.0000 - val_loss: 1.6288 - val_accuracy: 0.6364
Epoch 161/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0499 - accuracy:
0.9848 - val_loss: 1.6409 - val_accuracy: 0.5909
Epoch 162/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0491 - accuracy:
1.0000 - val_loss: 1.6370 - val_accuracy: 0.5909
Epoch 163/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0487 - accuracy:
1.0000 - val_loss: 1.6360 - val_accuracy: 0.6364
Epoch 164/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0512 - accuracy:
0.9848 - val_loss: 1.6613 - val_accuracy: 0.5909
Epoch 165/300
```

```
4/4 [==============================] - 0s 20ms/step - loss: 0.0462 - accuracy:
1.0000 - val_loss: 1.6637 - val_accuracy: 0.5909
Epoch 166/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0473 - accuracy:
1.0000 - val_loss: 1.6751 - val_accuracy: 0.5455
Epoch 167/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0458 - accuracy:
1.0000 - val_loss: 1.6692 - val_accuracy: 0.5455
Epoch 168/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0478 - accuracy:
1.0000 - val_loss: 1.6764 - val_accuracy: 0.5455
Epoch 169/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0465 - accuracy:
1.0000 - val_loss: 1.6681 - val_accuracy: 0.5909
Epoch 170/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0459 - accuracy:
1.0000 - val_loss: 1.7027 - val_accuracy: 0.5000
Epoch 171/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0468 - accuracy:
1.0000 - val_loss: 1.6789 - val_accuracy: 0.5455
Epoch 172/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0464 - accuracy:
1.0000 - val_loss: 1.6737 - val_accuracy: 0.5909
Epoch 173/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0462 - accuracy:
1.0000 - val_loss: 1.6843 - val_accuracy: 0.5909
Epoch 174/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0461 - accuracy:
1.0000 - val_loss: 1.6875 - val_accuracy: 0.5455
Epoch 175/300
4/4 [==============================] - 0s 21ms/step - loss: 0.0455 - accuracy:
1.0000 - val_loss: 1.7206 - val_accuracy: 0.5000
Epoch 176/300
4/4 [==============================] - 0s 16ms/step - loss: 0.0469 - accuracy:
1.0000 - val_loss: 1.7046 - val_accuracy: 0.5455
Epoch 177/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0450 - accuracy:
1.0000 - val_loss: 1.6915 - val_accuracy: 0.5455
Epoch 178/300
4/4 [==============================] - 0s 21ms/step - loss: 0.0450 - accuracy:
1.0000 - val_loss: 1.7243 - val_accuracy: 0.5000
Epoch 179/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0454 - accuracy:
1.0000 - val_loss: 1.7090 - val_accuracy: 0.5000
Epoch 180/300
4/4 [==============================] - 0s 16ms/step - loss: 0.0430 - accuracy:
1.0000 - val_loss: 1.6895 - val_accuracy: 0.5455
Epoch 181/300
```

```
4/4 [==============================] - 0s 21ms/step - loss: 0.0426 - accuracy:
1.0000 - val_loss: 1.6869 - val_accuracy: 0.5455
Epoch 182/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0432 - accuracy:
1.0000 - val_loss: 1.6852 - val_accuracy: 0.5455
Epoch 183/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0428 - accuracy:
1.0000 - val_loss: 1.6756 - val_accuracy: 0.5909
Epoch 184/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0415 - accuracy:
1.0000 - val_loss: 1.6870 - val_accuracy: 0.5909
Epoch 185/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0428 - accuracy:
1.0000 - val_loss: 1.6635 - val_accuracy: 0.5909
Epoch 186/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0438 - accuracy:
1.0000 - val_loss: 1.6661 - val_accuracy: 0.5909
Epoch 187/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0415 - accuracy:
1.0000 - val_loss: 1.6751 - val_accuracy: 0.5909
Epoch 188/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0416 - accuracy:
1.0000 - val_loss: 1.6686 - val_accuracy: 0.5909
Epoch 189/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0432 - accuracy:
1.0000 - val_loss: 1.6915 - val_accuracy: 0.5909
Epoch 190/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0413 - accuracy:
1.0000 - val_loss: 1.7125 - val_accuracy: 0.5909
Epoch 191/300
4/4 [==============================] - 0s 13ms/step - loss: 0.0411 - accuracy:
1.0000 - val_loss: 1.7056 - val_accuracy: 0.5909
Epoch 192/300
4/4 [==============================] - 0s 24ms/step - loss: 0.0408 - accuracy:
1.0000 - val_loss: 1.7023 - val_accuracy: 0.5909
Epoch 193/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0405 - accuracy:
1.0000 - val_loss: 1.7029 - val_accuracy: 0.5909
Epoch 194/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0400 - accuracy:
1.0000 - val_loss: 1.6893 - val_accuracy: 0.5909
Epoch 195/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0422 - accuracy:
1.0000 - val_loss: 1.7145 - val_accuracy: 0.5909
Epoch 196/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0400 - accuracy:
1.0000 - val_loss: 1.7238 - val_accuracy: 0.5455
Epoch 197/300
```

```
4/4 [==============================] - 0s 19ms/step - loss: 0.0399 - accuracy:
1.0000 - val_loss: 1.7075 - val_accuracy: 0.5909
Epoch 198/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0394 - accuracy:
1.0000 - val_loss: 1.6982 - val_accuracy: 0.5909
Epoch 199/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0394 - accuracy:
1.0000 - val_loss: 1.7082 - val_accuracy: 0.5909
Epoch 200/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0381 - accuracy:
1.0000 - val_loss: 1.7146 - val_accuracy: 0.5455
Epoch 201/300
4/4 [==============================] - 0s 13ms/step - loss: 0.0384 - accuracy:
1.0000 - val_loss: 1.7187 - val_accuracy: 0.5455
Epoch 202/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0374 - accuracy:
1.0000 - val_loss: 1.7195 - val_accuracy: 0.5455
Epoch 203/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0383 - accuracy:
1.0000 - val_loss: 1.7633 - val_accuracy: 0.5000
Epoch 204/300
4/4 [==============================] - 0s 13ms/step - loss: 0.0393 - accuracy:
1.0000 - val_loss: 1.7507 - val_accuracy: 0.5000
Epoch 205/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0386 - accuracy:
1.0000 - val_loss: 1.7050 - val_accuracy: 0.5909
Epoch 206/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0402 - accuracy:
1.0000 - val_loss: 1.7152 - val_accuracy: 0.5909
Epoch 207/300
4/4 [==============================] - 0s 24ms/step - loss: 0.0372 - accuracy:
1.0000 - val_loss: 1.7247 - val_accuracy: 0.5909
Epoch 208/300
4/4 [==============================] - 0s 32ms/step - loss: 0.0372 - accuracy:
1.0000 - val_loss: 1.7514 - val_accuracy: 0.5455
Epoch 209/300
4/4 [==============================] - 0s 23ms/step - loss: 0.0373 - accuracy:
1.0000 - val_loss: 1.7885 - val_accuracy: 0.5000
Epoch 210/300
4/4 [==============================] - 0s 22ms/step - loss: 0.0400 - accuracy:
1.0000 - val_loss: 1.7618 - val_accuracy: 0.5000
Epoch 211/300
4/4 [==============================] - 0s 25ms/step - loss: 0.0372 - accuracy:
1.0000 - val_loss: 1.7609 - val_accuracy: 0.5000
Epoch 212/300
4/4 [==============================] - 0s 23ms/step - loss: 0.0373 - accuracy:
1.0000 - val_loss: 1.7621 - val_accuracy: 0.5455
Epoch 213/300
```

```
4/4 [==============================] - 0s 23ms/step - loss: 0.0354 - accuracy:
1.0000 - val_loss: 1.7554 - val_accuracy: 0.5455
Epoch 214/300
4/4 [==============================] - 0s 17ms/step - loss: 0.0360 - accuracy:
1.0000 - val_loss: 1.7676 - val_accuracy: 0.5455
Epoch 215/300
4/4 [==============================] - 0s 22ms/step - loss: 0.0348 - accuracy:
1.0000 - val_loss: 1.7652 - val_accuracy: 0.5455
Epoch 216/300
4/4 [==============================] - 0s 22ms/step - loss: 0.0354 - accuracy:
1.0000 - val_loss: 1.7661 - val_accuracy: 0.5455
Epoch 217/300
4/4 [==============================] - 0s 22ms/step - loss: 0.0345 - accuracy:
1.0000 - val_loss: 1.7541 - val_accuracy: 0.5455
Epoch 218/300
4/4 [==============================] - 0s 18ms/step - loss: 0.0354 - accuracy:
1.0000 - val_loss: 1.7530 - val_accuracy: 0.5909
Epoch 219/300
4/4 [==============================] - 0s 17ms/step - loss: 0.0348 - accuracy:
1.0000 - val_loss: 1.7588 - val_accuracy: 0.5455
Epoch 220/300
4/4 [==============================] - 0s 30ms/step - loss: 0.0346 - accuracy:
1.0000 - val_loss: 1.7652 - val_accuracy: 0.5455
Epoch 221/300
4/4 [==============================] - 0s 28ms/step - loss: 0.0336 - accuracy:
1.0000 - val_loss: 1.7471 - val_accuracy: 0.5909
Epoch 222/300
4/4 [==============================] - 0s 22ms/step - loss: 0.0349 - accuracy:
1.0000 - val_loss: 1.7574 - val_accuracy: 0.5909
Epoch 223/300
4/4 [==============================] - 0s 23ms/step - loss: 0.0333 - accuracy:
1.0000 - val_loss: 1.7704 - val_accuracy: 0.5455
Epoch 224/300
4/4 [==============================] - 0s 25ms/step - loss: 0.0350 - accuracy:
1.0000 - val_loss: 1.7641 - val_accuracy: 0.5455
Epoch 225/300
4/4 [==============================] - 0s 23ms/step - loss: 0.0330 - accuracy:
1.0000 - val_loss: 1.7750 - val_accuracy: 0.5455
Epoch 226/300
4/4 [==============================] - 0s 25ms/step - loss: 0.0329 - accuracy:
1.0000 - val_loss: 1.7533 - val_accuracy: 0.5455
Epoch 227/300
4/4 [==============================] - 0s 28ms/step - loss: 0.0335 - accuracy:
1.0000 - val_loss: 1.7582 - val_accuracy: 0.5455
Epoch 228/300
4/4 [==============================] - 0s 28ms/step - loss: 0.0335 - accuracy:
1.0000 - val_loss: 1.7528 - val_accuracy: 0.5455
Epoch 229/300
```

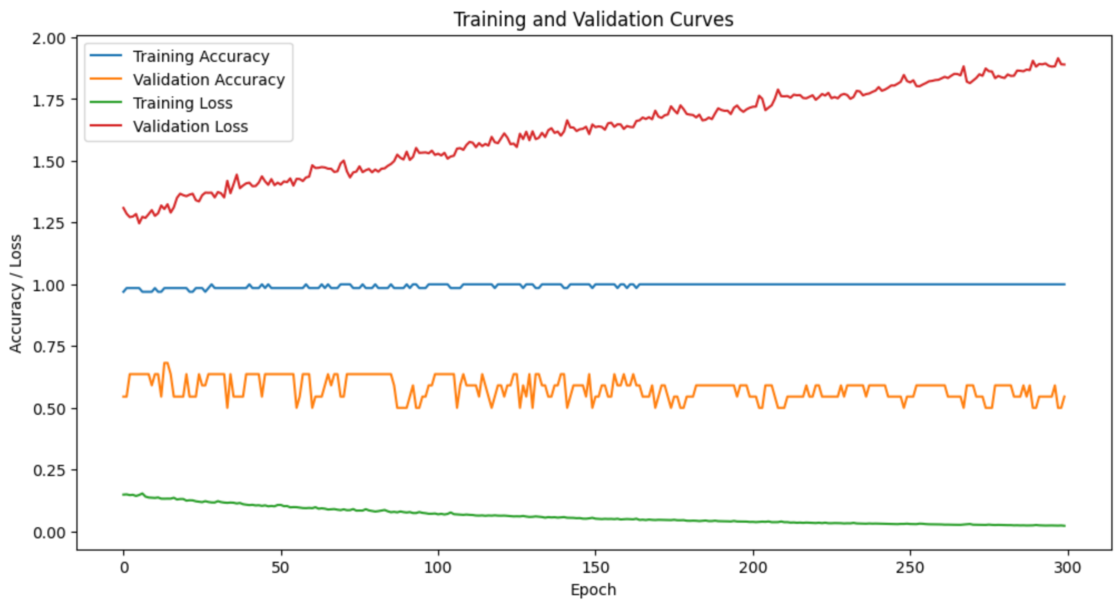
```
4/4 [==============================] - 0s 30ms/step - loss: 0.0331 - accuracy:
1.0000 - val_loss: 1.7663 - val_accuracy: 0.5909
Epoch 230/300
4/4 [==============================] - 0s 26ms/step - loss: 0.0320 - accuracy:
1.0000 - val_loss: 1.7707 - val_accuracy: 0.5455
Epoch 231/300
4/4 [==============================] - 0s 27ms/step - loss: 0.0326 - accuracy:
1.0000 - val_loss: 1.7661 - val_accuracy: 0.5909
Epoch 232/300
4/4 [==============================] - 0s 26ms/step - loss: 0.0324 - accuracy:
1.0000 - val_loss: 1.7508 - val_accuracy: 0.5909
Epoch 233/300
4/4 [==============================] - 0s 22ms/step - loss: 0.0343 - accuracy:
1.0000 - val_loss: 1.7582 - val_accuracy: 0.5909
Epoch 234/300
4/4 [==============================] - 0s 22ms/step - loss: 0.0320 - accuracy:
1.0000 - val_loss: 1.7830 - val_accuracy: 0.5909
Epoch 235/300
4/4 [==============================] - 0s 25ms/step - loss: 0.0317 - accuracy:
1.0000 - val_loss: 1.7810 - val_accuracy: 0.5909
Epoch 236/300
4/4 [==============================] - 0s 27ms/step - loss: 0.0311 - accuracy:
1.0000 - val_loss: 1.7649 - val_accuracy: 0.5909
Epoch 237/300
4/4 [==============================] - 0s 22ms/step - loss: 0.0309 - accuracy:
1.0000 - val_loss: 1.7699 - val_accuracy: 0.5455
Epoch 238/300
4/4 [==============================] - 0s 24ms/step - loss: 0.0319 - accuracy:
1.0000 - val_loss: 1.7724 - val_accuracy: 0.5909
Epoch 239/300
4/4 [==============================] - 0s 26ms/step - loss: 0.0312 - accuracy:
1.0000 - val_loss: 1.7790 - val_accuracy: 0.5909
Epoch 240/300
4/4 [==============================] - 0s 26ms/step - loss: 0.0315 - accuracy:
1.0000 - val_loss: 1.7842 - val_accuracy: 0.5909
Epoch 241/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0313 - accuracy:
1.0000 - val_loss: 1.7984 - val_accuracy: 0.5909
Epoch 242/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0309 - accuracy:
1.0000 - val_loss: 1.7834 - val_accuracy: 0.5909
Epoch 243/300
4/4 [==============================] - 0s 13ms/step - loss: 0.0306 - accuracy:
1.0000 - val_loss: 1.7884 - val_accuracy: 0.5909
Epoch 244/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0307 - accuracy:
1.0000 - val_loss: 1.7958 - val_accuracy: 0.5455
Epoch 245/300
```

```
4/4 [==============================] - 0s 15ms/step - loss: 0.0304 - accuracy:
1.0000 - val_loss: 1.8049 - val_accuracy: 0.5455
Epoch 246/300
4/4 [==============================] - 0s 21ms/step - loss: 0.0294 - accuracy:
1.0000 - val_loss: 1.8046 - val_accuracy: 0.5455
Epoch 247/300
4/4 [==============================] - 0s 22ms/step - loss: 0.0292 - accuracy:
1.0000 - val_loss: 1.8118 - val_accuracy: 0.5455
Epoch 248/300
4/4 [==============================] - 0s 13ms/step - loss: 0.0301 - accuracy:
1.0000 - val_loss: 1.8198 - val_accuracy: 0.5455
Epoch 249/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0301 - accuracy:
1.0000 - val_loss: 1.8474 - val_accuracy: 0.5000
Epoch 250/300
4/4 [==============================] - 0s 16ms/step - loss: 0.0310 - accuracy:
1.0000 - val_loss: 1.8230 - val_accuracy: 0.5455
Epoch 251/300
4/4 [==============================] - 0s 18ms/step - loss: 0.0297 - accuracy:
1.0000 - val_loss: 1.8173 - val_accuracy: 0.5455
Epoch 252/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0295 - accuracy:
1.0000 - val_loss: 1.8259 - val_accuracy: 0.5455
Epoch 253/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0293 - accuracy:
1.0000 - val_loss: 1.8025 - val_accuracy: 0.5909
Epoch 254/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0312 - accuracy:
1.0000 - val_loss: 1.8017 - val_accuracy: 0.5909
Epoch 255/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0306 - accuracy:
1.0000 - val_loss: 1.8086 - val_accuracy: 0.5909
Epoch 256/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0294 - accuracy:
1.0000 - val_loss: 1.8147 - val_accuracy: 0.5909
Epoch 257/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0287 - accuracy:
1.0000 - val_loss: 1.8218 - val_accuracy: 0.5909
Epoch 258/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0286 - accuracy:
1.0000 - val_loss: 1.8235 - val_accuracy: 0.5909
Epoch 259/300
4/4 [==============================] - 0s 16ms/step - loss: 0.0285 - accuracy:
1.0000 - val_loss: 1.8267 - val_accuracy: 0.5909
Epoch 260/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0282 - accuracy:
1.0000 - val_loss: 1.8278 - val_accuracy: 0.5909
Epoch 261/300
```

```
4/4 [==============================] - 0s 15ms/step - loss: 0.0279 - accuracy:
1.0000 - val_loss: 1.8341 - val_accuracy: 0.5909
Epoch 262/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0271 - accuracy:
1.0000 - val_loss: 1.8400 - val_accuracy: 0.5909
Epoch 263/300
4/4 [==============================] - 0s 22ms/step - loss: 0.0272 - accuracy:
1.0000 - val_loss: 1.8349 - val_accuracy: 0.5455
Epoch 264/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0267 - accuracy:
1.0000 - val_loss: 1.8439 - val_accuracy: 0.5455
Epoch 265/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0266 - accuracy:
1.0000 - val_loss: 1.8518 - val_accuracy: 0.5455
Epoch 266/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0264 - accuracy:
1.0000 - val_loss: 1.8513 - val_accuracy: 0.5455
Epoch 267/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0263 - accuracy:
1.0000 - val_loss: 1.8458 - val_accuracy: 0.5455
Epoch 268/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0273 - accuracy:
1.0000 - val_loss: 1.8824 - val_accuracy: 0.5000
Epoch 269/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0287 - accuracy:
1.0000 - val_loss: 1.8204 - val_accuracy: 0.5909
Epoch 270/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0298 - accuracy:
1.0000 - val_loss: 1.8143 - val_accuracy: 0.5909
Epoch 271/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0269 - accuracy:
1.0000 - val_loss: 1.8247 - val_accuracy: 0.5909
Epoch 272/300
4/4 [==============================] - 0s 22ms/step - loss: 0.0267 - accuracy:
1.0000 - val_loss: 1.8352 - val_accuracy: 0.5455
Epoch 273/300
4/4 [==============================] - 0s 21ms/step - loss: 0.0261 - accuracy:
1.0000 - val_loss: 1.8502 - val_accuracy: 0.5455
Epoch 274/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0260 - accuracy:
1.0000 - val_loss: 1.8441 - val_accuracy: 0.5455
Epoch 275/300
4/4 [==============================] - 0s 21ms/step - loss: 0.0259 - accuracy:
1.0000 - val_loss: 1.8737 - val_accuracy: 0.5000
Epoch 276/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0272 - accuracy:
1.0000 - val_loss: 1.8626 - val_accuracy: 0.5000
Epoch 277/300
```

```
4/4 [==============================] - 0s 21ms/step - loss: 0.0261 - accuracy:
1.0000 - val_loss: 1.8615 - val_accuracy: 0.5000
Epoch 278/300
4/4 [==============================] - 0s 21ms/step - loss: 0.0256 - accuracy:
1.0000 - val_loss: 1.8345 - val_accuracy: 0.5909
Epoch 279/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0261 - accuracy:
1.0000 - val_loss: 1.8437 - val_accuracy: 0.5909
Epoch 280/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0257 - accuracy:
1.0000 - val_loss: 1.8382 - val_accuracy: 0.5909
Epoch 281/300
4/4 [==============================] - 0s 21ms/step - loss: 0.0248 - accuracy:
1.0000 - val_loss: 1.8352 - val_accuracy: 0.5909
Epoch 282/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0251 - accuracy:
1.0000 - val_loss: 1.8494 - val_accuracy: 0.5909
Epoch 283/300
4/4 [==============================] - 0s 19ms/step - loss: 0.0252 - accuracy:
1.0000 - val_loss: 1.8435 - val_accuracy: 0.5909
Epoch 284/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0243 - accuracy:
1.0000 - val_loss: 1.8443 - val_accuracy: 0.5455
Epoch 285/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0246 - accuracy:
1.0000 - val_loss: 1.8652 - val_accuracy: 0.5455
Epoch 286/300
4/4 [==============================] - 0s 21ms/step - loss: 0.0241 - accuracy:
1.0000 - val_loss: 1.8643 - val_accuracy: 0.5455
Epoch 287/300
4/4 [==============================] - 0s 14ms/step - loss: 0.0244 - accuracy:
1.0000 - val_loss: 1.8623 - val_accuracy: 0.5909
Epoch 288/300
4/4 [==============================] - 0s 21ms/step - loss: 0.0237 - accuracy:
1.0000 - val_loss: 1.8692 - val_accuracy: 0.5455
Epoch 289/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0242 - accuracy:
1.0000 - val_loss: 1.8670 - val_accuracy: 0.5909
Epoch 290/300
4/4 [==============================] - 0s 21ms/step - loss: 0.0242 - accuracy:
1.0000 - val_loss: 1.9050 - val_accuracy: 0.5000
Epoch 291/300
4/4 [==============================] - 0s 16ms/step - loss: 0.0257 - accuracy:
1.0000 - val_loss: 1.8812 - val_accuracy: 0.5000
Epoch 292/300
4/4 [==============================] - 0s 18ms/step - loss: 0.0244 - accuracy:
1.0000 - val_loss: 1.8919 - val_accuracy: 0.5455
Epoch 293/300
```

```
4/4 [==============================] - 0s 16ms/step - loss: 0.0238 - accuracy:
1.0000 - val_loss: 1.8903 - val_accuracy: 0.5455
Epoch 294/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0234 - accuracy:
1.0000 - val_loss: 1.8940 - val_accuracy: 0.5455
Epoch 295/300
4/4 [==============================] - 0s 20ms/step - loss: 0.0238 - accuracy:
1.0000 - val_loss: 1.8847 - val_accuracy: 0.5455
Epoch 296/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0238 - accuracy:
1.0000 - val_loss: 1.8812 - val_accuracy: 0.5455
Epoch 297/300
4/4 [==============================] - 0s 18ms/step - loss: 0.0233 - accuracy:
1.0000 - val_loss: 1.8825 - val_accuracy: 0.5909
Epoch 298/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0232 - accuracy:
1.0000 - val_loss: 1.9153 - val_accuracy: 0.5000
Epoch 299/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0238 - accuracy:
1.0000 - val_loss: 1.8905 - val_accuracy: 0.5000
Epoch 300/300
4/4 [==============================] - 0s 15ms/step - loss: 0.0225 - accuracy:
1.0000 - val_loss: 1.8897 - val_accuracy: 0.5455
```



[77]:
```
# Save keras model to ONNX file

from aimodelshare.aimsonnx import model_to_onnx
```

```
onnx_model = model_to_onnx(keras_model, framework='keras',
                          transfer_learning=False,
                          deep_learning=True)

with open("model.onnx", "wb") as f:
    f.write(onnx_model.SerializeToString())
```

[78]:
```
# Submit keras model:

#-- Generate predicted y values

#Note: Keras predict returns the predicted column index location for␣
 ↪classification models
y_pred = keras_model.predict(preprocessor(X_test))
prediction_column_index = np.argmax(y_pred, axis=1) # Predict

# extract correct prediction labels
prediction_labels = [y_train.columns[i] for i in prediction_column_index]

model_filepath5 = "model.onnx"

# Submit Model 1 to Competition Leaderboard
mycompetition.submit_model(model = model_filepath5,
                                 prediction_submission=prediction_labels,
                                 preprocessor=preprocessor_filepath)
```

```
3/3 [==============================] - 0s 4ms/step
Insert search tags to help users find your model (optional): QMSS
Provide any useful notes about your model (optional): Qiankun Li

Your model has been submitted as model version 1134

To submit code used to create this model or to view current leaderboard navigate
to Model Playground:

 https://www.modelshare.ai/detail/model:3164
```

[79]:
```
# Check leaderboard

data = mycompetition.get_leaderboard()
mycompetition.stylize_leaderboard(data)
```

[79]:
```
<pandas.io.formats.style.Styler at 0x7bf025032860>
```

[80]:
```
## You are encouraged to try more experimentation and any other models by␣
 ↪adding more code cells to this notebook:
```

```
feature_count=X_train_processed.shape[1]#count features in input data
from keras.regularizers import l2

keras_model2 = Sequential()
keras_model2.add(Dense(256, input_dim=feature_count, activation='relu',␣
 ↪kernel_regularizer=l2(0.001)))
keras_model2.add(Dense(128, input_dim=feature_count, activation='relu'))
keras_model2.add(Dense(64, activation='relu'))
keras_model2.add(Dense(64, activation='relu'))
keras_model2.add(Dense(32, activation='relu'))## Define a Neural Network Model␣
 ↪with 5 layers 128->64->64->32->(?)


#Use Softmax activation in last layer. How many neurons should there be in the␣
 ↪last layer?
num_classes = 5
keras_model2.add(Dense(num_classes, activation='softmax'))


# Compile model
keras_model2.compile(loss='categorical_crossentropy', optimizer='sgd',␣
 ↪metrics=['accuracy'])

# Fitting the NN to the Training set
keras_model2.fit(preprocessor(X_train), y_train, ## Note that keras models␣
 ↪require a one-hot-encoded y_train object
               batch_size = 10,
               epochs = 300, validation_split=0.25)

y_pred2 = keras_model2.predict(preprocessor(X_test))
prediction_column_index2 = np.argmax(y_pred2, axis=1) # Predict

# extract correct prediction labels
prediction_labels2 = [y_train.columns[i] for i in prediction_column_index2]

model_filepath5 = "model.onnx"

mycompetition.submit_model(model = model_filepath5,
                                 prediction_submission=prediction_labels2,
                                 preprocessor=preprocessor_filepath)
## You can also try to import any new dataset pertaining to countries, merge␣
 ↪it, and see if it helps the predictions.
## If it does not, try to explain why it wasn't helpful by exploring variable␣
 ↪relationships.
```

```
Epoch 1/300
7/7 [==============================] - 1s 48ms/step - loss: 1.6831 - accuracy:
```

```
0.2121 - val_loss: 1.6798 - val_accuracy: 0.0909
Epoch 2/300
7/7 [==============================] - 0s 8ms/step - loss: 1.6696 - accuracy:
0.2273 - val_loss: 1.6756 - val_accuracy: 0.0909
Epoch 3/300
7/7 [==============================] - 0s 9ms/step - loss: 1.6584 - accuracy:
0.2879 - val_loss: 1.6718 - val_accuracy: 0.1364
Epoch 4/300
7/7 [==============================] - 0s 9ms/step - loss: 1.6488 - accuracy:
0.3030 - val_loss: 1.6692 - val_accuracy: 0.1818
Epoch 5/300
7/7 [==============================] - 0s 11ms/step - loss: 1.6405 - accuracy:
0.3788 - val_loss: 1.6667 - val_accuracy: 0.1818
Epoch 6/300
7/7 [==============================] - 0s 10ms/step - loss: 1.6334 - accuracy:
0.3636 - val_loss: 1.6647 - val_accuracy: 0.1818
Epoch 7/300
7/7 [==============================] - 0s 11ms/step - loss: 1.6267 - accuracy:
0.4091 - val_loss: 1.6629 - val_accuracy: 0.1818
Epoch 8/300
7/7 [==============================] - 0s 11ms/step - loss: 1.6199 - accuracy:
0.4545 - val_loss: 1.6609 - val_accuracy: 0.2273
Epoch 9/300
7/7 [==============================] - 0s 8ms/step - loss: 1.6143 - accuracy:
0.4394 - val_loss: 1.6590 - val_accuracy: 0.2273
Epoch 10/300
7/7 [==============================] - 0s 8ms/step - loss: 1.6076 - accuracy:
0.4394 - val_loss: 1.6572 - val_accuracy: 0.2273
Epoch 11/300
7/7 [==============================] - 0s 8ms/step - loss: 1.6011 - accuracy:
0.4242 - val_loss: 1.6557 - val_accuracy: 0.2273
Epoch 12/300
7/7 [==============================] - 0s 10ms/step - loss: 1.5938 - accuracy:
0.4091 - val_loss: 1.6536 - val_accuracy: 0.2273
Epoch 13/300
7/7 [==============================] - 0s 11ms/step - loss: 1.5891 - accuracy:
0.4242 - val_loss: 1.6504 - val_accuracy: 0.2273
Epoch 14/300
7/7 [==============================] - 0s 9ms/step - loss: 1.5812 - accuracy:
0.4242 - val_loss: 1.6476 - val_accuracy: 0.2273
Epoch 15/300
7/7 [==============================] - 0s 8ms/step - loss: 1.5746 - accuracy:
0.4394 - val_loss: 1.6448 - val_accuracy: 0.2727
Epoch 16/300
7/7 [==============================] - 0s 9ms/step - loss: 1.5671 - accuracy:
0.4394 - val_loss: 1.6417 - val_accuracy: 0.2727
Epoch 17/300
7/7 [==============================] - 0s 8ms/step - loss: 1.5600 - accuracy:
```

```
0.4394 - val_loss: 1.6386 - val_accuracy: 0.2727
Epoch 18/300
7/7 [==============================] - 0s 8ms/step - loss: 1.5522 - accuracy:
0.4697 - val_loss: 1.6349 - val_accuracy: 0.2727
Epoch 19/300
7/7 [==============================] - 0s 8ms/step - loss: 1.5438 - accuracy:
0.4697 - val_loss: 1.6311 - val_accuracy: 0.2727
Epoch 20/300
7/7 [==============================] - 0s 11ms/step - loss: 1.5348 - accuracy:
0.5000 - val_loss: 1.6258 - val_accuracy: 0.2727
Epoch 21/300
7/7 [==============================] - 0s 8ms/step - loss: 1.5263 - accuracy:
0.5000 - val_loss: 1.6196 - val_accuracy: 0.2727
Epoch 22/300
7/7 [==============================] - 0s 11ms/step - loss: 1.5161 - accuracy:
0.5000 - val_loss: 1.6133 - val_accuracy: 0.2727
Epoch 23/300
7/7 [==============================] - 0s 11ms/step - loss: 1.5057 - accuracy:
0.5000 - val_loss: 1.6054 - val_accuracy: 0.3182
Epoch 24/300
7/7 [==============================] - 0s 8ms/step - loss: 1.4940 - accuracy:
0.4848 - val_loss: 1.5985 - val_accuracy: 0.3182
Epoch 25/300
7/7 [==============================] - 0s 11ms/step - loss: 1.4817 - accuracy:
0.5000 - val_loss: 1.5911 - val_accuracy: 0.3182
Epoch 26/300
7/7 [==============================] - 0s 8ms/step - loss: 1.4676 - accuracy:
0.5000 - val_loss: 1.5828 - val_accuracy: 0.3182
Epoch 27/300
7/7 [==============================] - 0s 8ms/step - loss: 1.4526 - accuracy:
0.5000 - val_loss: 1.5730 - val_accuracy: 0.3182
Epoch 28/300
7/7 [==============================] - 0s 9ms/step - loss: 1.4375 - accuracy:
0.5000 - val_loss: 1.5619 - val_accuracy: 0.3182
Epoch 29/300
7/7 [==============================] - 0s 8ms/step - loss: 1.4224 - accuracy:
0.5000 - val_loss: 1.5515 - val_accuracy: 0.3182
Epoch 30/300
7/7 [==============================] - 0s 11ms/step - loss: 1.4056 - accuracy:
0.5000 - val_loss: 1.5387 - val_accuracy: 0.3182
Epoch 31/300
7/7 [==============================] - 0s 9ms/step - loss: 1.3884 - accuracy:
0.5000 - val_loss: 1.5256 - val_accuracy: 0.3182
Epoch 32/300
7/7 [==============================] - 0s 9ms/step - loss: 1.3695 - accuracy:
0.5152 - val_loss: 1.5131 - val_accuracy: 0.3182
Epoch 33/300
7/7 [==============================] - 0s 9ms/step - loss: 1.3494 - accuracy:
```

```
0.5152 - val_loss: 1.4992 - val_accuracy: 0.3182
Epoch 34/300
7/7 [==============================] - 0s 11ms/step - loss: 1.3327 - accuracy:
0.5303 - val_loss: 1.4861 - val_accuracy: 0.3182
Epoch 35/300
7/7 [==============================] - 0s 8ms/step - loss: 1.3126 - accuracy:
0.5152 - val_loss: 1.4692 - val_accuracy: 0.3182
Epoch 36/300
7/7 [==============================] - 0s 9ms/step - loss: 1.2945 - accuracy:
0.5455 - val_loss: 1.4559 - val_accuracy: 0.3636
Epoch 37/300
7/7 [==============================] - 0s 11ms/step - loss: 1.2744 - accuracy:
0.5758 - val_loss: 1.4390 - val_accuracy: 0.3636
Epoch 38/300
7/7 [==============================] - 0s 12ms/step - loss: 1.2549 - accuracy:
0.5606 - val_loss: 1.4238 - val_accuracy: 0.3636
Epoch 39/300
7/7 [==============================] - 0s 11ms/step - loss: 1.2349 - accuracy:
0.5758 - val_loss: 1.4095 - val_accuracy: 0.3636
Epoch 40/300
7/7 [==============================] - 0s 8ms/step - loss: 1.2174 - accuracy:
0.5758 - val_loss: 1.3984 - val_accuracy: 0.3636
Epoch 41/300
7/7 [==============================] - 0s 10ms/step - loss: 1.1949 - accuracy:
0.5909 - val_loss: 1.3849 - val_accuracy: 0.3636
Epoch 42/300
7/7 [==============================] - 0s 8ms/step - loss: 1.1795 - accuracy:
0.5909 - val_loss: 1.3677 - val_accuracy: 0.3636
Epoch 43/300
7/7 [==============================] - 0s 8ms/step - loss: 1.1611 - accuracy:
0.5909 - val_loss: 1.3504 - val_accuracy: 0.3636
Epoch 44/300
7/7 [==============================] - 0s 12ms/step - loss: 1.1411 - accuracy:
0.6061 - val_loss: 1.3346 - val_accuracy: 0.4091
Epoch 45/300
7/7 [==============================] - 0s 10ms/step - loss: 1.1214 - accuracy:
0.6212 - val_loss: 1.3181 - val_accuracy: 0.4091
Epoch 46/300
7/7 [==============================] - 0s 11ms/step - loss: 1.1026 - accuracy:
0.6212 - val_loss: 1.3054 - val_accuracy: 0.4091
Epoch 47/300
7/7 [==============================] - 0s 8ms/step - loss: 1.0856 - accuracy:
0.6212 - val_loss: 1.2839 - val_accuracy: 0.4091
Epoch 48/300
7/7 [==============================] - 0s 8ms/step - loss: 1.0636 - accuracy:
0.6212 - val_loss: 1.2701 - val_accuracy: 0.4091
Epoch 49/300
7/7 [==============================] - 0s 9ms/step - loss: 1.0451 - accuracy:
```

```
0.6212 - val_loss: 1.2527 - val_accuracy: 0.4091
Epoch 50/300
7/7 [==============================] - 0s 8ms/step - loss: 1.0262 - accuracy:
0.6515 - val_loss: 1.2380 - val_accuracy: 0.4091
Epoch 51/300
7/7 [==============================] - 0s 10ms/step - loss: 1.0059 - accuracy:
0.6818 - val_loss: 1.2205 - val_accuracy: 0.4545
Epoch 52/300
7/7 [==============================] - 0s 11ms/step - loss: 0.9881 - accuracy:
0.6970 - val_loss: 1.2043 - val_accuracy: 0.4545
Epoch 53/300
7/7 [==============================] - 0s 12ms/step - loss: 0.9655 - accuracy:
0.6818 - val_loss: 1.1791 - val_accuracy: 0.4545
Epoch 54/300
7/7 [==============================] - 0s 9ms/step - loss: 0.9457 - accuracy:
0.6970 - val_loss: 1.1602 - val_accuracy: 0.4545
Epoch 55/300
7/7 [==============================] - 0s 10ms/step - loss: 0.9268 - accuracy:
0.7121 - val_loss: 1.1489 - val_accuracy: 0.4545
Epoch 56/300
7/7 [==============================] - 0s 9ms/step - loss: 0.9114 - accuracy:
0.7121 - val_loss: 1.1378 - val_accuracy: 0.5000
Epoch 57/300
7/7 [==============================] - 0s 11ms/step - loss: 0.8961 - accuracy:
0.7424 - val_loss: 1.1281 - val_accuracy: 0.5000
Epoch 58/300
7/7 [==============================] - 0s 8ms/step - loss: 0.8743 - accuracy:
0.7424 - val_loss: 1.1097 - val_accuracy: 0.5000
Epoch 59/300
7/7 [==============================] - 0s 11ms/step - loss: 0.8651 - accuracy:
0.7727 - val_loss: 1.1096 - val_accuracy: 0.5000
Epoch 60/300
7/7 [==============================] - 0s 10ms/step - loss: 0.8474 - accuracy:
0.7424 - val_loss: 1.0934 - val_accuracy: 0.5000
Epoch 61/300
7/7 [==============================] - 0s 11ms/step - loss: 0.8328 - accuracy:
0.7424 - val_loss: 1.0844 - val_accuracy: 0.5000
Epoch 62/300
7/7 [==============================] - 0s 10ms/step - loss: 0.8165 - accuracy:
0.7727 - val_loss: 1.0710 - val_accuracy: 0.5000
Epoch 63/300
7/7 [==============================] - 0s 11ms/step - loss: 0.7983 - accuracy:
0.7727 - val_loss: 1.0566 - val_accuracy: 0.5000
Epoch 64/300
7/7 [==============================] - 0s 11ms/step - loss: 0.7883 - accuracy:
0.7576 - val_loss: 1.0460 - val_accuracy: 0.5455
Epoch 65/300
7/7 [==============================] - 0s 11ms/step - loss: 0.7793 - accuracy:
```

```
0.7879 - val_loss: 1.0609 - val_accuracy: 0.5000
Epoch 66/300
7/7 [==============================] - 0s 9ms/step - loss: 0.7601 - accuracy:
0.8333 - val_loss: 1.0272 - val_accuracy: 0.5909
Epoch 67/300
7/7 [==============================] - 0s 9ms/step - loss: 0.7471 - accuracy:
0.8182 - val_loss: 1.0345 - val_accuracy: 0.5455
Epoch 68/300
7/7 [==============================] - 0s 11ms/step - loss: 0.7367 - accuracy:
0.8182 - val_loss: 1.0173 - val_accuracy: 0.5909
Epoch 69/300
7/7 [==============================] - 0s 9ms/step - loss: 0.7233 - accuracy:
0.8485 - val_loss: 1.0203 - val_accuracy: 0.5455
Epoch 70/300
7/7 [==============================] - 0s 8ms/step - loss: 0.7076 - accuracy:
0.8333 - val_loss: 1.0019 - val_accuracy: 0.5455
Epoch 71/300
7/7 [==============================] - 0s 11ms/step - loss: 0.6973 - accuracy:
0.8333 - val_loss: 0.9913 - val_accuracy: 0.5909
Epoch 72/300
7/7 [==============================] - 0s 9ms/step - loss: 0.6784 - accuracy:
0.8333 - val_loss: 0.9719 - val_accuracy: 0.5909
Epoch 73/300
7/7 [==============================] - 0s 9ms/step - loss: 0.6666 - accuracy:
0.8485 - val_loss: 0.9759 - val_accuracy: 0.6364
Epoch 74/300
7/7 [==============================] - 0s 9ms/step - loss: 0.6599 - accuracy:
0.8485 - val_loss: 0.9607 - val_accuracy: 0.5455
Epoch 75/300
7/7 [==============================] - 0s 14ms/step - loss: 0.6527 - accuracy:
0.8485 - val_loss: 0.9515 - val_accuracy: 0.6364
Epoch 76/300
7/7 [==============================] - 0s 9ms/step - loss: 0.6389 - accuracy:
0.8636 - val_loss: 0.9484 - val_accuracy: 0.5455
Epoch 77/300
7/7 [==============================] - 0s 9ms/step - loss: 0.6249 - accuracy:
0.8485 - val_loss: 0.9315 - val_accuracy: 0.6364
Epoch 78/300
7/7 [==============================] - 0s 10ms/step - loss: 0.6110 - accuracy:
0.8636 - val_loss: 0.9739 - val_accuracy: 0.5455
Epoch 79/300
7/7 [==============================] - 0s 10ms/step - loss: 0.6096 - accuracy:
0.8485 - val_loss: 0.9746 - val_accuracy: 0.5000
Epoch 80/300
7/7 [==============================] - 0s 12ms/step - loss: 0.5942 - accuracy:
0.8636 - val_loss: 0.9368 - val_accuracy: 0.5455
Epoch 81/300
7/7 [==============================] - 0s 13ms/step - loss: 0.5732 - accuracy:
```

```
0.8636 - val_loss: 0.9162 - val_accuracy: 0.6364
Epoch 82/300
7/7 [==============================] - 0s 11ms/step - loss: 0.5760 - accuracy:
0.8636 - val_loss: 0.9308 - val_accuracy: 0.5909
Epoch 83/300
7/7 [==============================] - 0s 11ms/step - loss: 0.5605 - accuracy:
0.8788 - val_loss: 0.9186 - val_accuracy: 0.6364
Epoch 84/300
7/7 [==============================] - 0s 11ms/step - loss: 0.5427 - accuracy:
0.8636 - val_loss: 0.9212 - val_accuracy: 0.6364
Epoch 85/300
7/7 [==============================] - 0s 11ms/step - loss: 0.5325 - accuracy:
0.8788 - val_loss: 0.9440 - val_accuracy: 0.5455
Epoch 86/300
7/7 [==============================] - 0s 11ms/step - loss: 0.5200 - accuracy:
0.8636 - val_loss: 0.9225 - val_accuracy: 0.5909
Epoch 87/300
7/7 [==============================] - 0s 9ms/step - loss: 0.5164 - accuracy:
0.8636 - val_loss: 0.9362 - val_accuracy: 0.5909
Epoch 88/300
7/7 [==============================] - 0s 11ms/step - loss: 0.5030 - accuracy:
0.8636 - val_loss: 0.8928 - val_accuracy: 0.6364
Epoch 89/300
7/7 [==============================] - 0s 8ms/step - loss: 0.4981 - accuracy:
0.8788 - val_loss: 0.8961 - val_accuracy: 0.5909
Epoch 90/300
7/7 [==============================] - 0s 8ms/step - loss: 0.4959 - accuracy:
0.8636 - val_loss: 0.9071 - val_accuracy: 0.5909
Epoch 91/300
7/7 [==============================] - 0s 11ms/step - loss: 0.4753 - accuracy:
0.8788 - val_loss: 0.9032 - val_accuracy: 0.6364
Epoch 92/300
7/7 [==============================] - 0s 11ms/step - loss: 0.4757 - accuracy:
0.8788 - val_loss: 0.9568 - val_accuracy: 0.5909
Epoch 93/300
7/7 [==============================] - 0s 10ms/step - loss: 0.4704 - accuracy:
0.8939 - val_loss: 0.9051 - val_accuracy: 0.6364
Epoch 94/300
7/7 [==============================] - 0s 11ms/step - loss: 0.4465 - accuracy:
0.8788 - val_loss: 0.9683 - val_accuracy: 0.6364
Epoch 95/300
7/7 [==============================] - 0s 13ms/step - loss: 0.4409 - accuracy:
0.9091 - val_loss: 0.9233 - val_accuracy: 0.5909
Epoch 96/300
7/7 [==============================] - 0s 9ms/step - loss: 0.4257 - accuracy:
0.8939 - val_loss: 0.9022 - val_accuracy: 0.5909
Epoch 97/300
7/7 [==============================] - 0s 15ms/step - loss: 0.4170 - accuracy:
```

```
0.9091 - val_loss: 0.8632 - val_accuracy: 0.7273
Epoch 98/300
7/7 [==============================] - 0s 14ms/step - loss: 0.4187 - accuracy:
0.9091 - val_loss: 0.9729 - val_accuracy: 0.5909
Epoch 99/300
7/7 [==============================] - 0s 15ms/step - loss: 0.4069 - accuracy:
0.8788 - val_loss: 0.9321 - val_accuracy: 0.6364
Epoch 100/300
7/7 [==============================] - 0s 13ms/step - loss: 0.3922 - accuracy:
0.9394 - val_loss: 0.9140 - val_accuracy: 0.6364
Epoch 101/300
7/7 [==============================] - 0s 13ms/step - loss: 0.3846 - accuracy:
0.9242 - val_loss: 0.9161 - val_accuracy: 0.5909
Epoch 102/300
7/7 [==============================] - 0s 15ms/step - loss: 0.3796 - accuracy:
0.9091 - val_loss: 0.9321 - val_accuracy: 0.6364
Epoch 103/300
7/7 [==============================] - 0s 12ms/step - loss: 0.3733 - accuracy:
0.9091 - val_loss: 0.8825 - val_accuracy: 0.7273
Epoch 104/300
7/7 [==============================] - 0s 14ms/step - loss: 0.3592 - accuracy:
0.9394 - val_loss: 0.8912 - val_accuracy: 0.6364
Epoch 105/300
7/7 [==============================] - 0s 15ms/step - loss: 0.3596 - accuracy:
0.8939 - val_loss: 0.9528 - val_accuracy: 0.5909
Epoch 106/300
7/7 [==============================] - 0s 15ms/step - loss: 0.3477 - accuracy:
0.9394 - val_loss: 0.9004 - val_accuracy: 0.6818
Epoch 107/300
7/7 [==============================] - 0s 12ms/step - loss: 0.3382 - accuracy:
0.9242 - val_loss: 0.9088 - val_accuracy: 0.6818
Epoch 108/300
7/7 [==============================] - 0s 14ms/step - loss: 0.3299 - accuracy:
0.9242 - val_loss: 0.9614 - val_accuracy: 0.5909
Epoch 109/300
7/7 [==============================] - 0s 13ms/step - loss: 0.3287 - accuracy:
0.9091 - val_loss: 0.8971 - val_accuracy: 0.6818
Epoch 110/300
7/7 [==============================] - 0s 14ms/step - loss: 0.3195 - accuracy:
0.9394 - val_loss: 0.9540 - val_accuracy: 0.6364
Epoch 111/300
7/7 [==============================] - 0s 13ms/step - loss: 0.3171 - accuracy:
0.9545 - val_loss: 0.9850 - val_accuracy: 0.5909
Epoch 112/300
7/7 [==============================] - 0s 16ms/step - loss: 0.3267 - accuracy:
0.9242 - val_loss: 0.9673 - val_accuracy: 0.6364
Epoch 113/300
7/7 [==============================] - 0s 12ms/step - loss: 0.3003 - accuracy:
```

```
0.9545 - val_loss: 0.9715 - val_accuracy: 0.6364
Epoch 114/300
7/7 [==============================] - 0s 14ms/step - loss: 0.2929 - accuracy:
0.9545 - val_loss: 0.9641 - val_accuracy: 0.7273
Epoch 115/300
7/7 [==============================] - 0s 12ms/step - loss: 0.2855 - accuracy:
0.9697 - val_loss: 0.9240 - val_accuracy: 0.7273
Epoch 116/300
7/7 [==============================] - 0s 14ms/step - loss: 0.2796 - accuracy:
0.9545 - val_loss: 0.9497 - val_accuracy: 0.6818
Epoch 117/300
7/7 [==============================] - 0s 15ms/step - loss: 0.2758 - accuracy:
0.9394 - val_loss: 0.9671 - val_accuracy: 0.6364
Epoch 118/300
7/7 [==============================] - 0s 14ms/step - loss: 0.2745 - accuracy:
0.9545 - val_loss: 0.9483 - val_accuracy: 0.6364
Epoch 119/300
7/7 [==============================] - 0s 14ms/step - loss: 0.2665 - accuracy:
0.9545 - val_loss: 0.9605 - val_accuracy: 0.6818
Epoch 120/300
7/7 [==============================] - 0s 15ms/step - loss: 0.2634 - accuracy:
0.9545 - val_loss: 0.9423 - val_accuracy: 0.6818
Epoch 121/300
7/7 [==============================] - 0s 14ms/step - loss: 0.2535 - accuracy:
0.9545 - val_loss: 0.9848 - val_accuracy: 0.6364
Epoch 122/300
7/7 [==============================] - 0s 13ms/step - loss: 0.2507 - accuracy:
0.9545 - val_loss: 0.9717 - val_accuracy: 0.5909
Epoch 123/300
7/7 [==============================] - 0s 15ms/step - loss: 0.2469 - accuracy:
0.9545 - val_loss: 0.9688 - val_accuracy: 0.6818
Epoch 124/300
7/7 [==============================] - 0s 14ms/step - loss: 0.2416 - accuracy:
0.9545 - val_loss: 0.9682 - val_accuracy: 0.6818
Epoch 125/300
7/7 [==============================] - 0s 9ms/step - loss: 0.2389 - accuracy:
0.9545 - val_loss: 1.0032 - val_accuracy: 0.6364
Epoch 126/300
7/7 [==============================] - 0s 11ms/step - loss: 0.2345 - accuracy:
0.9697 - val_loss: 0.9539 - val_accuracy: 0.6818
Epoch 127/300
7/7 [==============================] - 0s 11ms/step - loss: 0.2371 - accuracy:
0.9545 - val_loss: 1.0021 - val_accuracy: 0.6364
Epoch 128/300
7/7 [==============================] - 0s 9ms/step - loss: 0.2242 - accuracy:
0.9545 - val_loss: 0.9728 - val_accuracy: 0.7273
Epoch 129/300
7/7 [==============================] - 0s 12ms/step - loss: 0.2214 - accuracy:
```

```
0.9545 - val_loss: 1.0031 - val_accuracy: 0.6818
Epoch 130/300
7/7 [==============================] - 0s 9ms/step - loss: 0.2137 - accuracy:
0.9697 - val_loss: 0.9969 - val_accuracy: 0.6818
Epoch 131/300
7/7 [==============================] - 0s 8ms/step - loss: 0.2256 - accuracy:
0.9545 - val_loss: 1.0376 - val_accuracy: 0.5909
Epoch 132/300
7/7 [==============================] - 0s 11ms/step - loss: 0.2137 - accuracy:
0.9545 - val_loss: 1.0123 - val_accuracy: 0.6364
Epoch 133/300
7/7 [==============================] - 0s 8ms/step - loss: 0.2024 - accuracy:
0.9697 - val_loss: 1.0049 - val_accuracy: 0.6818
Epoch 134/300
7/7 [==============================] - 0s 11ms/step - loss: 0.2017 - accuracy:
0.9697 - val_loss: 0.9900 - val_accuracy: 0.6818
Epoch 135/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1964 - accuracy:
0.9848 - val_loss: 1.0254 - val_accuracy: 0.7273
Epoch 136/300
7/7 [==============================] - 0s 8ms/step - loss: 0.1964 - accuracy:
0.9545 - val_loss: 1.0546 - val_accuracy: 0.6364
Epoch 137/300
7/7 [==============================] - 0s 11ms/step - loss: 0.2004 - accuracy:
0.9697 - val_loss: 1.0363 - val_accuracy: 0.6364
Epoch 138/300
7/7 [==============================] - 0s 8ms/step - loss: 0.1918 - accuracy:
0.9697 - val_loss: 1.0642 - val_accuracy: 0.6364
Epoch 139/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1869 - accuracy:
0.9848 - val_loss: 1.0871 - val_accuracy: 0.6364
Epoch 140/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1851 - accuracy:
0.9697 - val_loss: 1.0437 - val_accuracy: 0.6364
Epoch 141/300
7/7 [==============================] - 0s 8ms/step - loss: 0.1837 - accuracy:
0.9545 - val_loss: 1.0805 - val_accuracy: 0.6364
Epoch 142/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1785 - accuracy:
0.9848 - val_loss: 1.0906 - val_accuracy: 0.6364
Epoch 143/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1774 - accuracy:
0.9697 - val_loss: 1.0504 - val_accuracy: 0.6818
Epoch 144/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1729 - accuracy:
0.9848 - val_loss: 1.0883 - val_accuracy: 0.6364
Epoch 145/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1779 - accuracy:
```

```
0.9697 - val_loss: 1.0890 - val_accuracy: 0.6364
Epoch 146/300
7/7 [==============================] - 0s 13ms/step - loss: 0.1672 - accuracy:
0.9848 - val_loss: 1.0652 - val_accuracy: 0.6818
Epoch 147/300
7/7 [==============================] - 0s 10ms/step - loss: 0.1674 - accuracy:
0.9848 - val_loss: 1.0500 - val_accuracy: 0.6818
Epoch 148/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1776 - accuracy:
0.9545 - val_loss: 1.0783 - val_accuracy: 0.6364
Epoch 149/300
7/7 [==============================] - 0s 10ms/step - loss: 0.1631 - accuracy:
0.9697 - val_loss: 1.0990 - val_accuracy: 0.6364
Epoch 150/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1645 - accuracy:
0.9848 - val_loss: 1.0786 - val_accuracy: 0.6818
Epoch 151/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1660 - accuracy:
0.9848 - val_loss: 1.1263 - val_accuracy: 0.6364
Epoch 152/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1609 - accuracy:
0.9697 - val_loss: 1.1024 - val_accuracy: 0.6364
Epoch 153/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1573 - accuracy:
0.9848 - val_loss: 1.0741 - val_accuracy: 0.6818
Epoch 154/300
7/7 [==============================] - 0s 12ms/step - loss: 0.1537 - accuracy:
0.9697 - val_loss: 1.1016 - val_accuracy: 0.6364
Epoch 155/300
7/7 [==============================] - 0s 8ms/step - loss: 0.1511 - accuracy:
0.9848 - val_loss: 1.1207 - val_accuracy: 0.6364
Epoch 156/300
7/7 [==============================] - 0s 12ms/step - loss: 0.1493 - accuracy:
0.9848 - val_loss: 1.1064 - val_accuracy: 0.7273
Epoch 157/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1462 - accuracy:
0.9848 - val_loss: 1.1405 - val_accuracy: 0.6364
Epoch 158/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1459 - accuracy:
0.9848 - val_loss: 1.1097 - val_accuracy: 0.7273
Epoch 159/300
7/7 [==============================] - 0s 13ms/step - loss: 0.1518 - accuracy:
0.9697 - val_loss: 1.1101 - val_accuracy: 0.7273
Epoch 160/300
7/7 [==============================] - 0s 8ms/step - loss: 0.1445 - accuracy:
0.9848 - val_loss: 1.1356 - val_accuracy: 0.5909
Epoch 161/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1377 - accuracy:
```

```
0.9848 - val_loss: 1.1251 - val_accuracy: 0.6364
Epoch 162/300
7/7 [==============================] - 0s 8ms/step - loss: 0.1386 - accuracy:
0.9848 - val_loss: 1.1439 - val_accuracy: 0.5909
Epoch 163/300
7/7 [==============================] - 0s 10ms/step - loss: 0.1399 - accuracy:
1.0000 - val_loss: 1.1583 - val_accuracy: 0.5909
Epoch 164/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1419 - accuracy:
0.9848 - val_loss: 1.1190 - val_accuracy: 0.5909
Epoch 165/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1378 - accuracy:
0.9848 - val_loss: 1.1501 - val_accuracy: 0.5909
Epoch 166/300
7/7 [==============================] - 0s 8ms/step - loss: 0.1351 - accuracy:
0.9848 - val_loss: 1.2015 - val_accuracy: 0.5909
Epoch 167/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1377 - accuracy:
0.9697 - val_loss: 1.1869 - val_accuracy: 0.5909
Epoch 168/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1288 - accuracy:
1.0000 - val_loss: 1.1877 - val_accuracy: 0.5455
Epoch 169/300
7/7 [==============================] - 0s 8ms/step - loss: 0.1374 - accuracy:
0.9848 - val_loss: 1.1619 - val_accuracy: 0.6364
Epoch 170/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1320 - accuracy:
0.9848 - val_loss: 1.2100 - val_accuracy: 0.5909
Epoch 171/300
7/7 [==============================] - 0s 12ms/step - loss: 0.1313 - accuracy:
0.9848 - val_loss: 1.1552 - val_accuracy: 0.6364
Epoch 172/300
7/7 [==============================] - 0s 10ms/step - loss: 0.1297 - accuracy:
0.9848 - val_loss: 1.1764 - val_accuracy: 0.5909
Epoch 173/300
7/7 [==============================] - 0s 10ms/step - loss: 0.1251 - accuracy:
1.0000 - val_loss: 1.1499 - val_accuracy: 0.6818
Epoch 174/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1249 - accuracy:
0.9848 - val_loss: 1.2252 - val_accuracy: 0.5909
Epoch 175/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1232 - accuracy:
0.9848 - val_loss: 1.2138 - val_accuracy: 0.5909
Epoch 176/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1205 - accuracy:
1.0000 - val_loss: 1.2319 - val_accuracy: 0.5909
Epoch 177/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1221 - accuracy:
```

```
0.9848 - val_loss: 1.2481 - val_accuracy: 0.5909
Epoch 178/300
7/7 [==============================] - 0s 13ms/step - loss: 0.1171 - accuracy:
1.0000 - val_loss: 1.2150 - val_accuracy: 0.5909
Epoch 179/300
7/7 [==============================] - 0s 10ms/step - loss: 0.1183 - accuracy:
1.0000 - val_loss: 1.2389 - val_accuracy: 0.5909
Epoch 180/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1140 - accuracy:
1.0000 - val_loss: 1.2559 - val_accuracy: 0.5909
Epoch 181/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1155 - accuracy:
1.0000 - val_loss: 1.3052 - val_accuracy: 0.5909
Epoch 182/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1201 - accuracy:
0.9848 - val_loss: 1.2735 - val_accuracy: 0.5909
Epoch 183/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1145 - accuracy:
1.0000 - val_loss: 1.2699 - val_accuracy: 0.5909
Epoch 184/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1141 - accuracy:
0.9848 - val_loss: 1.2188 - val_accuracy: 0.6364
Epoch 185/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1126 - accuracy:
0.9848 - val_loss: 1.2806 - val_accuracy: 0.5909
Epoch 186/300
7/7 [==============================] - 0s 14ms/step - loss: 0.1123 - accuracy:
1.0000 - val_loss: 1.2772 - val_accuracy: 0.5909
Epoch 187/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1141 - accuracy:
1.0000 - val_loss: 1.2836 - val_accuracy: 0.5909
Epoch 188/300
7/7 [==============================] - 0s 10ms/step - loss: 0.1124 - accuracy:
0.9848 - val_loss: 1.2585 - val_accuracy: 0.5909
Epoch 189/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1102 - accuracy:
1.0000 - val_loss: 1.2532 - val_accuracy: 0.5909
Epoch 190/300
7/7 [==============================] - 0s 10ms/step - loss: 0.1073 - accuracy:
1.0000 - val_loss: 1.2973 - val_accuracy: 0.5909
Epoch 191/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1074 - accuracy:
1.0000 - val_loss: 1.2258 - val_accuracy: 0.6818
Epoch 192/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1127 - accuracy:
0.9848 - val_loss: 1.2532 - val_accuracy: 0.6364
Epoch 193/300
7/7 [==============================] - 0s 12ms/step - loss: 0.1045 - accuracy:
```

```
1.0000 - val_loss: 1.3046 - val_accuracy: 0.5909
Epoch 194/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1023 - accuracy:
1.0000 - val_loss: 1.2714 - val_accuracy: 0.5909
Epoch 195/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1041 - accuracy:
1.0000 - val_loss: 1.2371 - val_accuracy: 0.6818
Epoch 196/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1094 - accuracy:
0.9848 - val_loss: 1.2892 - val_accuracy: 0.5909
Epoch 197/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1002 - accuracy:
1.0000 - val_loss: 1.2908 - val_accuracy: 0.5909
Epoch 198/300
7/7 [==============================] - 0s 10ms/step - loss: 0.1024 - accuracy:
1.0000 - val_loss: 1.2472 - val_accuracy: 0.6364
Epoch 199/300
7/7 [==============================] - 0s 9ms/step - loss: 0.1019 - accuracy:
1.0000 - val_loss: 1.3754 - val_accuracy: 0.5909
Epoch 200/300
7/7 [==============================] - 0s 16ms/step - loss: 0.1037 - accuracy:
1.0000 - val_loss: 1.2916 - val_accuracy: 0.5909
Epoch 201/300
7/7 [==============================] - 0s 11ms/step - loss: 0.1023 - accuracy:
1.0000 - val_loss: 1.2967 - val_accuracy: 0.5909
Epoch 202/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0990 - accuracy:
1.0000 - val_loss: 1.3401 - val_accuracy: 0.5909
Epoch 203/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0967 - accuracy:
1.0000 - val_loss: 1.3309 - val_accuracy: 0.5909
Epoch 204/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0994 - accuracy:
1.0000 - val_loss: 1.3404 - val_accuracy: 0.5455
Epoch 205/300
7/7 [==============================] - 0s 8ms/step - loss: 0.0992 - accuracy:
1.0000 - val_loss: 1.3200 - val_accuracy: 0.5909
Epoch 206/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0992 - accuracy:
1.0000 - val_loss: 1.3251 - val_accuracy: 0.5909
Epoch 207/300
7/7 [==============================] - 0s 12ms/step - loss: 0.0970 - accuracy:
1.0000 - val_loss: 1.3567 - val_accuracy: 0.5909
Epoch 208/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0977 - accuracy:
1.0000 - val_loss: 1.3834 - val_accuracy: 0.5909
Epoch 209/300
7/7 [==============================] - 0s 12ms/step - loss: 0.0971 - accuracy:
```

```
1.0000 - val_loss: 1.3682 - val_accuracy: 0.5909
Epoch 210/300
7/7 [==============================] - 0s 8ms/step - loss: 0.0957 - accuracy:
1.0000 - val_loss: 1.3833 - val_accuracy: 0.5909
Epoch 211/300
7/7 [==============================] - 0s 8ms/step - loss: 0.0992 - accuracy:
1.0000 - val_loss: 1.3925 - val_accuracy: 0.5909
Epoch 212/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0955 - accuracy:
1.0000 - val_loss: 1.3427 - val_accuracy: 0.5909
Epoch 213/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0952 - accuracy:
1.0000 - val_loss: 1.3968 - val_accuracy: 0.5909
Epoch 214/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0954 - accuracy:
1.0000 - val_loss: 1.3767 - val_accuracy: 0.5909
Epoch 215/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0936 - accuracy:
1.0000 - val_loss: 1.4008 - val_accuracy: 0.5909
Epoch 216/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0912 - accuracy:
1.0000 - val_loss: 1.3894 - val_accuracy: 0.5909
Epoch 217/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0916 - accuracy:
1.0000 - val_loss: 1.3843 - val_accuracy: 0.5909
Epoch 218/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0915 - accuracy:
1.0000 - val_loss: 1.3713 - val_accuracy: 0.5909
Epoch 219/300
7/7 [==============================] - 0s 12ms/step - loss: 0.0921 - accuracy:
1.0000 - val_loss: 1.3626 - val_accuracy: 0.5909
Epoch 220/300
7/7 [==============================] - 0s 12ms/step - loss: 0.0916 - accuracy:
1.0000 - val_loss: 1.3723 - val_accuracy: 0.5909
Epoch 221/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0888 - accuracy:
1.0000 - val_loss: 1.3961 - val_accuracy: 0.5909
Epoch 222/300
7/7 [==============================] - 0s 10ms/step - loss: 0.0903 - accuracy:
1.0000 - val_loss: 1.4343 - val_accuracy: 0.5909
Epoch 223/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0901 - accuracy:
1.0000 - val_loss: 1.4205 - val_accuracy: 0.5909
Epoch 224/300
7/7 [==============================] - 0s 10ms/step - loss: 0.0891 - accuracy:
1.0000 - val_loss: 1.3811 - val_accuracy: 0.5909
Epoch 225/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0889 - accuracy:
```

```
1.0000 - val_loss: 1.4121 - val_accuracy: 0.5909
Epoch 226/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0887 - accuracy:
1.0000 - val_loss: 1.3936 - val_accuracy: 0.5909
Epoch 227/300
7/7 [==============================] - 0s 14ms/step - loss: 0.0882 - accuracy:
1.0000 - val_loss: 1.3847 - val_accuracy: 0.5909
Epoch 228/300
7/7 [==============================] - 0s 12ms/step - loss: 0.0862 - accuracy:
1.0000 - val_loss: 1.4161 - val_accuracy: 0.5909
Epoch 229/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0878 - accuracy:
1.0000 - val_loss: 1.4417 - val_accuracy: 0.5909
Epoch 230/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0854 - accuracy:
1.0000 - val_loss: 1.4242 - val_accuracy: 0.5909
Epoch 231/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0865 - accuracy:
1.0000 - val_loss: 1.4769 - val_accuracy: 0.5909
Epoch 232/300
7/7 [==============================] - 0s 12ms/step - loss: 0.0876 - accuracy:
1.0000 - val_loss: 1.3782 - val_accuracy: 0.6364
Epoch 233/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0886 - accuracy:
1.0000 - val_loss: 1.4007 - val_accuracy: 0.5909
Epoch 234/300
7/7 [==============================] - 0s 12ms/step - loss: 0.0871 - accuracy:
1.0000 - val_loss: 1.4375 - val_accuracy: 0.5909
Epoch 235/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0861 - accuracy:
1.0000 - val_loss: 1.4241 - val_accuracy: 0.5909
Epoch 236/300
7/7 [==============================] - 0s 8ms/step - loss: 0.0864 - accuracy:
1.0000 - val_loss: 1.4026 - val_accuracy: 0.6364
Epoch 237/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0867 - accuracy:
1.0000 - val_loss: 1.4618 - val_accuracy: 0.5909
Epoch 238/300
7/7 [==============================] - 0s 10ms/step - loss: 0.0840 - accuracy:
1.0000 - val_loss: 1.4380 - val_accuracy: 0.5909
Epoch 239/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0842 - accuracy:
1.0000 - val_loss: 1.4477 - val_accuracy: 0.5909
Epoch 240/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0831 - accuracy:
1.0000 - val_loss: 1.5207 - val_accuracy: 0.5909
Epoch 241/300
7/7 [==============================] - 0s 10ms/step - loss: 0.0852 - accuracy:
```

```
1.0000 - val_loss: 1.4795 - val_accuracy: 0.5909
Epoch 242/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0829 - accuracy:
1.0000 - val_loss: 1.4564 - val_accuracy: 0.5909
Epoch 243/300
7/7 [==============================] - 0s 10ms/step - loss: 0.0831 - accuracy:
1.0000 - val_loss: 1.4715 - val_accuracy: 0.5909
Epoch 244/300
7/7 [==============================] - 0s 13ms/step - loss: 0.0818 - accuracy:
1.0000 - val_loss: 1.5192 - val_accuracy: 0.5909
Epoch 245/300
7/7 [==============================] - 0s 8ms/step - loss: 0.0825 - accuracy:
1.0000 - val_loss: 1.4420 - val_accuracy: 0.5909
Epoch 246/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0824 - accuracy:
1.0000 - val_loss: 1.4826 - val_accuracy: 0.5909
Epoch 247/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0813 - accuracy:
1.0000 - val_loss: 1.4994 - val_accuracy: 0.5909
Epoch 248/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0813 - accuracy:
1.0000 - val_loss: 1.5154 - val_accuracy: 0.5909
Epoch 249/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0812 - accuracy:
1.0000 - val_loss: 1.4855 - val_accuracy: 0.5909
Epoch 250/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0801 - accuracy:
1.0000 - val_loss: 1.5167 - val_accuracy: 0.5909
Epoch 251/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0791 - accuracy:
1.0000 - val_loss: 1.5066 - val_accuracy: 0.5909
Epoch 252/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0789 - accuracy:
1.0000 - val_loss: 1.5004 - val_accuracy: 0.5909
Epoch 253/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0796 - accuracy:
1.0000 - val_loss: 1.4785 - val_accuracy: 0.5909
Epoch 254/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0792 - accuracy:
1.0000 - val_loss: 1.5119 - val_accuracy: 0.5909
Epoch 255/300
7/7 [==============================] - 0s 12ms/step - loss: 0.0791 - accuracy:
1.0000 - val_loss: 1.4883 - val_accuracy: 0.5909
Epoch 256/300
7/7 [==============================] - 0s 8ms/step - loss: 0.0787 - accuracy:
1.0000 - val_loss: 1.5082 - val_accuracy: 0.5909
Epoch 257/300
7/7 [==============================] - 0s 8ms/step - loss: 0.0785 - accuracy:
```

```
1.0000 - val_loss: 1.5265 - val_accuracy: 0.5909
Epoch 258/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0789 - accuracy:
1.0000 - val_loss: 1.5333 - val_accuracy: 0.5909
Epoch 259/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0783 - accuracy:
1.0000 - val_loss: 1.5362 - val_accuracy: 0.5909
Epoch 260/300
7/7 [==============================] - 0s 14ms/step - loss: 0.0783 - accuracy:
1.0000 - val_loss: 1.5108 - val_accuracy: 0.5909
Epoch 261/300
7/7 [==============================] - 0s 13ms/step - loss: 0.0782 - accuracy:
1.0000 - val_loss: 1.5024 - val_accuracy: 0.5909
Epoch 262/300
7/7 [==============================] - 0s 15ms/step - loss: 0.0778 - accuracy:
1.0000 - val_loss: 1.5180 - val_accuracy: 0.5909
Epoch 263/300
7/7 [==============================] - 0s 16ms/step - loss: 0.0775 - accuracy:
1.0000 - val_loss: 1.5159 - val_accuracy: 0.5909
Epoch 264/300
7/7 [==============================] - 0s 13ms/step - loss: 0.0770 - accuracy:
1.0000 - val_loss: 1.5459 - val_accuracy: 0.5909
Epoch 265/300
7/7 [==============================] - 0s 13ms/step - loss: 0.0775 - accuracy:
1.0000 - val_loss: 1.5455 - val_accuracy: 0.5909
Epoch 266/300
7/7 [==============================] - 0s 14ms/step - loss: 0.0771 - accuracy:
1.0000 - val_loss: 1.5258 - val_accuracy: 0.5909
Epoch 267/300
7/7 [==============================] - 0s 16ms/step - loss: 0.0767 - accuracy:
1.0000 - val_loss: 1.5466 - val_accuracy: 0.5909
Epoch 268/300
7/7 [==============================] - 0s 13ms/step - loss: 0.0756 - accuracy:
1.0000 - val_loss: 1.5294 - val_accuracy: 0.5909
Epoch 269/300
7/7 [==============================] - 0s 17ms/step - loss: 0.0760 - accuracy:
1.0000 - val_loss: 1.5503 - val_accuracy: 0.5909
Epoch 270/300
7/7 [==============================] - 0s 14ms/step - loss: 0.0764 - accuracy:
1.0000 - val_loss: 1.5379 - val_accuracy: 0.5909
Epoch 271/300
7/7 [==============================] - 0s 12ms/step - loss: 0.0751 - accuracy:
1.0000 - val_loss: 1.5484 - val_accuracy: 0.5909
Epoch 272/300
7/7 [==============================] - 0s 13ms/step - loss: 0.0750 - accuracy:
1.0000 - val_loss: 1.5449 - val_accuracy: 0.5909
Epoch 273/300
7/7 [==============================] - 0s 13ms/step - loss: 0.0758 - accuracy:
```

```
1.0000 - val_loss: 1.5549 - val_accuracy: 0.5909
Epoch 274/300
7/7 [==============================] - 0s 13ms/step - loss: 0.0756 - accuracy:
1.0000 - val_loss: 1.5598 - val_accuracy: 0.5909
Epoch 275/300
7/7 [==============================] - 0s 14ms/step - loss: 0.0753 - accuracy:
1.0000 - val_loss: 1.5666 - val_accuracy: 0.5909
Epoch 276/300
7/7 [==============================] - 0s 14ms/step - loss: 0.0751 - accuracy:
1.0000 - val_loss: 1.5669 - val_accuracy: 0.5909
Epoch 277/300
7/7 [==============================] - 0s 15ms/step - loss: 0.0747 - accuracy:
1.0000 - val_loss: 1.5916 - val_accuracy: 0.5909
Epoch 278/300
7/7 [==============================] - 0s 16ms/step - loss: 0.0754 - accuracy:
1.0000 - val_loss: 1.5518 - val_accuracy: 0.5909
Epoch 279/300
7/7 [==============================] - 0s 15ms/step - loss: 0.0745 - accuracy:
1.0000 - val_loss: 1.5555 - val_accuracy: 0.5909
Epoch 280/300
7/7 [==============================] - 0s 12ms/step - loss: 0.0746 - accuracy:
1.0000 - val_loss: 1.5328 - val_accuracy: 0.5909
Epoch 281/300
7/7 [==============================] - 0s 14ms/step - loss: 0.0752 - accuracy:
1.0000 - val_loss: 1.5569 - val_accuracy: 0.5909
Epoch 282/300
7/7 [==============================] - 0s 15ms/step - loss: 0.0741 - accuracy:
1.0000 - val_loss: 1.5817 - val_accuracy: 0.5909
Epoch 283/300
7/7 [==============================] - 0s 15ms/step - loss: 0.0739 - accuracy:
1.0000 - val_loss: 1.5999 - val_accuracy: 0.5909
Epoch 284/300
7/7 [==============================] - 0s 13ms/step - loss: 0.0743 - accuracy:
1.0000 - val_loss: 1.5914 - val_accuracy: 0.5909
Epoch 285/300
7/7 [==============================] - 0s 17ms/step - loss: 0.0742 - accuracy:
1.0000 - val_loss: 1.5849 - val_accuracy: 0.5909
Epoch 286/300
7/7 [==============================] - 0s 13ms/step - loss: 0.0736 - accuracy:
1.0000 - val_loss: 1.5906 - val_accuracy: 0.5909
Epoch 287/300
7/7 [==============================] - 0s 18ms/step - loss: 0.0734 - accuracy:
1.0000 - val_loss: 1.5837 - val_accuracy: 0.5909
Epoch 288/300
7/7 [==============================] - 0s 14ms/step - loss: 0.0734 - accuracy:
1.0000 - val_loss: 1.5683 - val_accuracy: 0.5909
Epoch 289/300
7/7 [==============================] - 0s 14ms/step - loss: 0.0738 - accuracy:
```

```
1.0000 - val_loss: 1.5980 - val_accuracy: 0.5909
Epoch 290/300
7/7 [==============================] - 0s 14ms/step - loss: 0.0732 - accuracy:
1.0000 - val_loss: 1.5900 - val_accuracy: 0.5909
Epoch 291/300
7/7 [==============================] - 0s 10ms/step - loss: 0.0728 - accuracy:
1.0000 - val_loss: 1.5909 - val_accuracy: 0.5909
Epoch 292/300
7/7 [==============================] - 0s 12ms/step - loss: 0.0728 - accuracy:
1.0000 - val_loss: 1.6024 - val_accuracy: 0.5909
Epoch 293/300
7/7 [==============================] - 0s 12ms/step - loss: 0.0722 - accuracy:
1.0000 - val_loss: 1.5935 - val_accuracy: 0.5909
Epoch 294/300
7/7 [==============================] - 0s 8ms/step - loss: 0.0724 - accuracy:
1.0000 - val_loss: 1.6129 - val_accuracy: 0.5909
Epoch 295/300
7/7 [==============================] - 0s 10ms/step - loss: 0.0723 - accuracy:
1.0000 - val_loss: 1.6064 - val_accuracy: 0.5909
Epoch 296/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0723 - accuracy:
1.0000 - val_loss: 1.5998 - val_accuracy: 0.5909
Epoch 297/300
7/7 [==============================] - 0s 11ms/step - loss: 0.0719 - accuracy:
1.0000 - val_loss: 1.6087 - val_accuracy: 0.5909
Epoch 298/300
7/7 [==============================] - 0s 13ms/step - loss: 0.0719 - accuracy:
1.0000 - val_loss: 1.6099 - val_accuracy: 0.5909
Epoch 299/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0718 - accuracy:
1.0000 - val_loss: 1.6040 - val_accuracy: 0.5909
Epoch 300/300
7/7 [==============================] - 0s 9ms/step - loss: 0.0720 - accuracy:
1.0000 - val_loss: 1.6024 - val_accuracy: 0.5909
3/3 [==============================] - 0s 4ms/step
Insert search tags to help users find your model (optional): QMSS
Provide any useful notes about your model (optional): Qiankun Li

Your model has been submitted as model version 1135

To submit code used to create this model or to view current leaderboard navigate
to Model Playground:

 https://www.modelshare.ai/detail/model:3164
```

8 7. Submission of final report and clean code to github

[This is a final project you display on your GitHub to the World]

Instructions - Make a new notebook, visualize any plots you found relevant - Reproduce the code you used for the best models and display results - Write what insights you found useful and what behaviours were observed - Make it in a style of a clean, succint report (within the .ipynb) - Upload this final report notebook to a new repository in your personal github account - Remember to paste the link of your final repo at the top of this notebook where asked

[80]:
```

```

Github Repo: