MTP_BNN

March 18, 2022

packages

[1]:
```
#empty
```

[2]:
```
!pip install pyforest
```

```
Collecting pyforest
  Downloading pyforest-1.1.0.tar.gz (15 kB)
Building wheels for collected packages: pyforest
  Building wheel for pyforest (setup.py) … done
  Created wheel for pyforest: filename=pyforest-1.1.0-py2.py3-none-any.whl
size=14607
sha256=a60c6e88ad47fe850fcce04c7aba1fd31c21b80c1a89b9f0246aa77029a2a2ab
  Stored in directory: /root/.cache/pip/wheels/61/1c/da/48e6c884142d485475d852d6
9d20a096aba5beceb338822893
Successfully built pyforest
Installing collected packages: pyforest
Successfully installed pyforest-1.1.0
```

[3]:
```
#automatic imports required packages as per usage in code
import pyforest
```

[4]:
```
#packages
!pip install tensorflow-probability
!pip install nbconvert
```

```
Requirement already satisfied: tensorflow-probability in
/usr/local/lib/python3.7/dist-packages (0.16.0)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-
packages (from tensorflow-probability) (1.21.5)
Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-
packages (from tensorflow-probability) (4.4.2)
Requirement already satisfied: absl-py in /usr/local/lib/python3.7/dist-packages
(from tensorflow-probability) (1.0.0)
Requirement already satisfied: dm-tree in /usr/local/lib/python3.7/dist-packages
(from tensorflow-probability) (0.1.6)
Requirement already satisfied: gast>=0.3.2 in /usr/local/lib/python3.7/dist-
packages (from tensorflow-probability) (0.5.3)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-
```

```
packages (from tensorflow-probability) (1.15.0)
Requirement already satisfied: cloudpickle>=1.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow-probability) (1.3.0)
Requirement already satisfied: nbconvert in /usr/local/lib/python3.7/dist-
packages (5.6.1)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.7/dist-
packages (from nbconvert) (5.1.1)
Requirement already satisfied: jinja2>=2.4 in /usr/local/lib/python3.7/dist-
packages (from nbconvert) (2.11.3)
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-
packages (from nbconvert) (2.6.1)
Requirement already satisfied: entrypoints>=0.2.2 in
/usr/local/lib/python3.7/dist-packages (from nbconvert) (0.4)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.7/dist-
packages (from nbconvert) (0.7.1)
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.7/dist-
packages (from nbconvert) (4.9.2)
Requirement already satisfied: nbformat>=4.4 in /usr/local/lib/python3.7/dist-
packages (from nbconvert) (5.1.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.7/dist-packages
(from nbconvert) (4.1.0)
Requirement already satisfied: mistune<2,>=0.8.1 in
/usr/local/lib/python3.7/dist-packages (from nbconvert) (0.8.4)
Requirement already satisfied: testpath in /usr/local/lib/python3.7/dist-
packages (from nbconvert) (0.6.0)
Requirement already satisfied: pandocfilters>=1.4.1 in
/usr/local/lib/python3.7/dist-packages (from nbconvert) (1.5.0)
Requirement already satisfied: MarkupSafe>=0.23 in
/usr/local/lib/python3.7/dist-packages (from jinja2>=2.4->nbconvert) (2.0.1)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in
/usr/local/lib/python3.7/dist-packages (from nbformat>=4.4->nbconvert) (4.3.3)
Requirement already satisfied: ipython-genutils in
/usr/local/lib/python3.7/dist-packages (from nbformat>=4.4->nbconvert) (0.2.0)
Requirement already satisfied: importlib-metadata in
/usr/local/lib/python3.7/dist-packages (from
jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (4.11.2)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from
jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (3.10.0.2)
Requirement already satisfied: importlib-resources>=1.4.0 in
/usr/local/lib/python3.7/dist-packages (from
jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (5.4.0)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.7/dist-
packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (21.4.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in
/usr/local/lib/python3.7/dist-packages (from
jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (0.18.1)
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.7/dist-
```

```
packages (from importlib-
resources>=1.4.0->jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (3.7.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-
packages (from bleach->nbconvert) (21.3)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-
packages (from bleach->nbconvert) (1.15.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.7/dist-
packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging->bleach->nbconvert)
(3.0.7)
```

[5]:
```
import pandas as pd
import numpy as np
```

0.0.1 DATA

import data

[6]:
```
#using official url to load data
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00601/ai4i2020.
↪csv'

data = pd.read_csv(url)

data.head()
```

[6]:
```
   UDI Product ID Type  Air temperature [K]  Process temperature [K]  \
0    1     M14860    M                298.1                    308.6
1    2     L47181    L                298.2                    308.7
2    3     L47182    L                298.1                    308.5
3    4     L47183    L                298.2                    308.6
4    5     L47184    L                298.2                    308.7

   Rotational speed [rpm]  Torque [Nm]  Tool wear [min]  Machine failure  TWF  \
0                    1551         42.8                0                0    0
1                    1408         46.3                3                0    0
2                    1498         49.4                5                0    0
3                    1433         39.5                7                0    0
4                    1408         40.0                9                0    0

   HDF  PWF  OSF  RNF
0    0    0    0    0
1    0    0    0    0
2    0    0    0    0
3    0    0    0    0
4    0    0    0    0
```

data description taken from UCI:

Abstract: The AI4I 2020 Predictive Maintenance Dataset is a synthetic dataset that reflects real predictive maintenance data encountered in industry.

Variable	Value
Data Set Characteristics:	Multivariate, Time-Series
Number of Instances:	10000
Area:	Computer
Attribute Characteristics:	Real
Number of Attributes:	14
Date Donated:	2020-08-30
Associated Tasks:	Classification, Regression, Causal-Discovery
Missing Values?	N/A
Number of Web Hits:	33135

** Data Set Information: **

Since real predictive maintenance datasets are generally difficult to obtain and in particular difficult to publish, we present and provide a synthetic dataset that reflects real predictive maintenance encountered in industry to the best of our knowledge.

Attribute Information:

The dataset consists of 10 000 data points stored as rows with 14 features in columns UID: unique identifier ranging from 1 to 10000 product ID: consisting of a letter L, M, or H for low (50% of all products), medium (30%) and high (20%) as product quality variants and a variant-specific serial number air temperature [K]: generated using a random walk process later normalized to a standard deviation of 2 K around 300 K process temperature [K]: generated using a random walk process normalized to a standard deviation of 1 K, added to the air temperature plus 10 K. rotational speed [rpm]: calculated from a power of 2860 W, overlaid with a normally distributed noise torque [Nm]: torque values are normally distributed around 40 Nm with a Ïf = 10 Nm and no negative values. tool wear [min]: The quality variants H/M/L add 5/3/2 minutes of tool wear to the used tool in the process. and a 'machine failure' label that indicates, whether the machine has failed in this particular datapoint for any of the following failure modes are true.

The machine failure consists of five independent failure modes tool wear failure (TWF): the tool will be replaced of fail at a randomly selected tool wear time between 200 â€“ 240 mins (120 times in our dataset). At this point in time, the tool is replaced 69 times, and fails 51 times (randomly assigned). heat dissipation failure (HDF): heat dissipation causes a process failure, if the difference between air- and process temperature is below 8.6 K and the toolâ€™s rotational speed is below 1380 rpm. This is the case for 115 data points. power failure (PWF): the product of torque and rotational speed (in rad/s) equals the power required for the process. If this power is below 3500 W or above 9000 W, the process fails, which is the case 95 times in our dataset. overstrain failure (OSF): if the product of tool wear and torque exceeds 11,000 minNm for the L product variant (12,000 M, 13,000 H), the process fails due to overstrain. This is true for 98 datapoints. random failures (RNF): each process has a chance of 0,1 % to fail regardless of its process parameters. This is the case for only 5 datapoints, less than could be expected for 10,000 datapoints in our dataset.

If at least one of the above failure modes is true, the process fails and the 'machine failure' label is set to 1. It is therefore not transparent to the machine learning method, which of the failure modes has caused the process to fail

Relevant Papers:

Stephan Matzka, 'Explainable Artificial Intelligence for Predictive Maintenance Applications', Third International Conference on Artificial Intelligence for Industries (AI4I 2020), 2020 (in press)

[7]:
```
data.describe()
```

[7]:
```
               UDI  Air temperature [K]  Process temperature [K]  \
count  10000.00000         10000.000000             10000.000000
mean    5000.50000           300.004930               310.005560
std     2886.89568             2.000259                 1.483734
min        1.00000           295.300000               305.700000
25%     2500.75000           298.300000               308.800000
50%     5000.50000           300.100000               310.100000
75%     7500.25000           301.500000               311.100000
max    10000.00000           304.500000               313.800000

       Rotational speed [rpm]   Torque [Nm]  Tool wear [min]  Machine failure  \
count            10000.000000  10000.000000     10000.000000     10000.000000
mean              1538.776100     39.986910       107.951000         0.033900
std                179.284096      9.968934        63.654147         0.180981
min               1168.000000      3.800000         0.000000         0.000000
25%               1423.000000     33.200000        53.000000         0.000000
50%               1503.000000     40.100000       108.000000         0.000000
75%               1612.000000     46.800000       162.000000         0.000000
max               2886.000000     76.600000       253.000000         1.000000

                TWF           HDF           PWF           OSF          RNF
count  10000.000000  10000.000000  10000.000000  10000.000000  10000.00000
mean       0.004600      0.011500      0.009500      0.009800      0.00190
std        0.067671      0.106625      0.097009      0.098514      0.04355
min        0.000000      0.000000      0.000000      0.000000      0.00000
25%        0.000000      0.000000      0.000000      0.000000      0.00000
50%        0.000000      0.000000      0.000000      0.000000      0.00000
75%        0.000000      0.000000      0.000000      0.000000      0.00000
max        1.000000      1.000000      1.000000      1.000000      1.00000
```

[8]:
```
#for i in data:
    #print(data[i].unique())
```

[9]:
```
data.nunique()
```

[9]:
```
UDI                        10000
Product ID                 10000
```

```
Type                           3
Air temperature [K]           93
Process temperature [K]       82
Rotational speed [rpm]       941
Torque [Nm]                  577
Tool wear [min]              246
Machine failure                2
TWF                            2
HDF                            2
PWF                            2
OSF                            2
RNF                            2
dtype: int64
```

[10]:

```python
#basic info about dataset
df = data
df.shape
df.index
df.columns
df.info()
df.count()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   UDI                      10000 non-null  int64
 1   Product ID               10000 non-null  object
 2   Type                     10000 non-null  object
 3   Air temperature [K]      10000 non-null  float64
 4   Process temperature [K]  10000 non-null  float64
 5   Rotational speed [rpm]   10000 non-null  int64
 6   Torque [Nm]              10000 non-null  float64
 7   Tool wear [min]          10000 non-null  int64
 8   Machine failure          10000 non-null  int64
 9   TWF                      10000 non-null  int64
 10  HDF                      10000 non-null  int64
 11  PWF                      10000 non-null  int64
 12  OSF                      10000 non-null  int64
 13  RNF                      10000 non-null  int64
dtypes: float64(3), int64(9), object(2)
memory usage: 1.1+ MB
```

[10]:

```
UDI                        10000
Product ID                 10000
Type                       10000
```

```
Air temperature [K]        10000
Process temperature [K]    10000
Rotational speed [rpm]     10000
Torque [Nm]                10000
Tool wear [min]            10000
Machine failure            10000
TWF                        10000
HDF                        10000
PWF                        10000
OSF                        10000
RNF                        10000
dtype: int64
```

[11]:

```
df.sum()
df.cumsum()
df.min()
df.max()
df.describe()
df.mean()
df.median()
```

```
/usr/local/lib/python3.7/dist-packages/pyforest/__init__.py:6: FutureWarning:
Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only valid
columns before calling the reduction.
  install_nbextension,
/usr/local/lib/python3.7/dist-packages/pyforest/__init__.py:7: FutureWarning:
Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only valid
columns before calling the reduction.
  install_labextension,
```

[11]:

```
UDI                        5000.5
Air temperature [K]         300.1
Process temperature [K]     310.1
Rotational speed [rpm]     1503.0
Torque [Nm]                  40.1
Tool wear [min]             108.0
Machine failure               0.0
TWF                           0.0
HDF                           0.0
PWF                           0.0
OSF                           0.0
RNF                           0.0
dtype: float64
```

preprocessing data

[12]:
```python
#define X and y from df
# product id is unique for each data row and its not important
# but we have product type of 3 categories
# L, M, H are three types representing for low (50% of all products),
# medium (30%) and high (20%) as product quality variants respectively
df['Type'].unique()
```

[12]:
```
array(['M', 'L', 'H'], dtype=object)
```

[13]:
```python
# converting this categorical data to numerical with class 0, 1, 2 for L,M,H␣
↪respectively
# using OrdinalEncoder from sklearn for ordinal data of product quality variant
# indicating l for low quality, m for medium quality, h for high quality
# one-hot encoding is not suitable for ordinal data
from sklearn.preprocessing import OrdinalEncoder
ordinal_encoder = OrdinalEncoder()
df['Type'] = ordinal_encoder.fit_transform(df[['Type']])
df['Type'].unique()
# this gives categories converted into integers
```

[13]:
```
array([2., 1., 0.])
```

[14]:
```python
# these are original categories in data
ordinal_encoder.categories_
```

[14]:
```
[array(['H', 'L', 'M'], dtype=object)]
```

[15]:
```python
# this sorts all the categories present and assigns values to them in␣
↪alphabetical order
# 0 for H
# 1 for L
# 2 for M
print(ordinal_encoder.inverse_transform([[0]]))
print(ordinal_encoder.inverse_transform([[1]]))
print(ordinal_encoder.inverse_transform([[2]]))
```

```
[['H']]
[['L']]
[['M']]
```

[16]:
```python
df.describe()
```

[16]:
```
                UDI         Type  Air temperature [K]  Process temperature [K]  \
count  10000.00000  10000.00000         10000.000000             10000.000000
mean    5000.50000      1.19940           300.004930               310.005560
std     2886.89568      0.60023             2.000259                 1.483734
min        1.00000      0.00000           295.300000               305.700000
```

```
25%     2500.75000      1.00000          298.300000               308.800000
50%     5000.50000      1.00000          300.100000               310.100000
75%     7500.25000      2.00000          301.500000               311.100000
max    10000.00000      2.00000          304.500000               313.800000

       Rotational speed [rpm]  Torque [Nm]  Tool wear [min]  Machine failure  \
count            10000.000000  10000.000000     10000.000000     10000.000000
mean              1538.776100     39.986910       107.951000         0.033900
std                179.284096      9.968934        63.654147         0.180981
min               1168.000000      3.800000         0.000000         0.000000
25%               1423.000000     33.200000        53.000000         0.000000
50%               1503.000000     40.100000       108.000000         0.000000
75%               1612.000000     46.800000       162.000000         0.000000
max               2886.000000     76.600000       253.000000         1.000000

                TWF           HDF           PWF           OSF          RNF
count  10000.000000  10000.000000  10000.000000  10000.000000  10000.00000
mean       0.004600      0.011500      0.009500      0.009800      0.00190
std        0.067671      0.106625      0.097009      0.098514      0.04355
min        0.000000      0.000000      0.000000      0.000000      0.00000
25%        0.000000      0.000000      0.000000      0.000000      0.00000
50%        0.000000      0.000000      0.000000      0.000000      0.00000
75%        0.000000      0.000000      0.000000      0.000000      0.00000
max        1.000000      1.000000      1.000000      1.000000      1.00000
```

[17]:
```
df.nunique()
```

[17]:
```
UDI                        10000
Product ID                 10000
Type                           3
Air temperature [K]           93
Process temperature [K]       82
Rotational speed [rpm]       941
Torque [Nm]                  577
Tool wear [min]              246
Machine failure                2
TWF                            2
HDF                            2
PWF                            2
OSF                            2
RNF                            2
dtype: int64
```

[18]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
```

```
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   UDI                      10000 non-null  int64
 1   Product ID               10000 non-null  object
 2   Type                     10000 non-null  float64
 3   Air temperature [K]      10000 non-null  float64
 4   Process temperature [K]  10000 non-null  float64
 5   Rotational speed [rpm]   10000 non-null  int64
 6   Torque [Nm]              10000 non-null  float64
 7   Tool wear [min]          10000 non-null  int64
 8   Machine failure          10000 non-null  int64
 9   TWF                      10000 non-null  int64
 10  HDF                      10000 non-null  int64
 11  PWF                      10000 non-null  int64
 12  OSF                      10000 non-null  int64
 13  RNF                      10000 non-null  int64
dtypes: float64(4), int64(9), object(1)
memory usage: 1.1+ MB
```

[19]:
```
# now make the final dataset to be used in NN
# remove the product id variable
# remaining attributes are of types either int64 or float64
df.drop('Product ID', axis=1, inplace=True)
df.drop('UDI', axis=1, inplace=True)
```

[20]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Type                     10000 non-null  float64
 1   Air temperature [K]      10000 non-null  float64
 2   Process temperature [K]  10000 non-null  float64
 3   Rotational speed [rpm]   10000 non-null  int64
 4   Torque [Nm]              10000 non-null  float64
 5   Tool wear [min]          10000 non-null  int64
 6   Machine failure          10000 non-null  int64
 7   TWF                      10000 non-null  int64
 8   HDF                      10000 non-null  int64
 9   PWF                      10000 non-null  int64
 10  OSF                      10000 non-null  int64
 11  RNF                      10000 non-null  int64
dtypes: float64(4), int64(8)
memory usage: 937.6 KB
```

[21]:
```
## add mitosheet data visualization
```

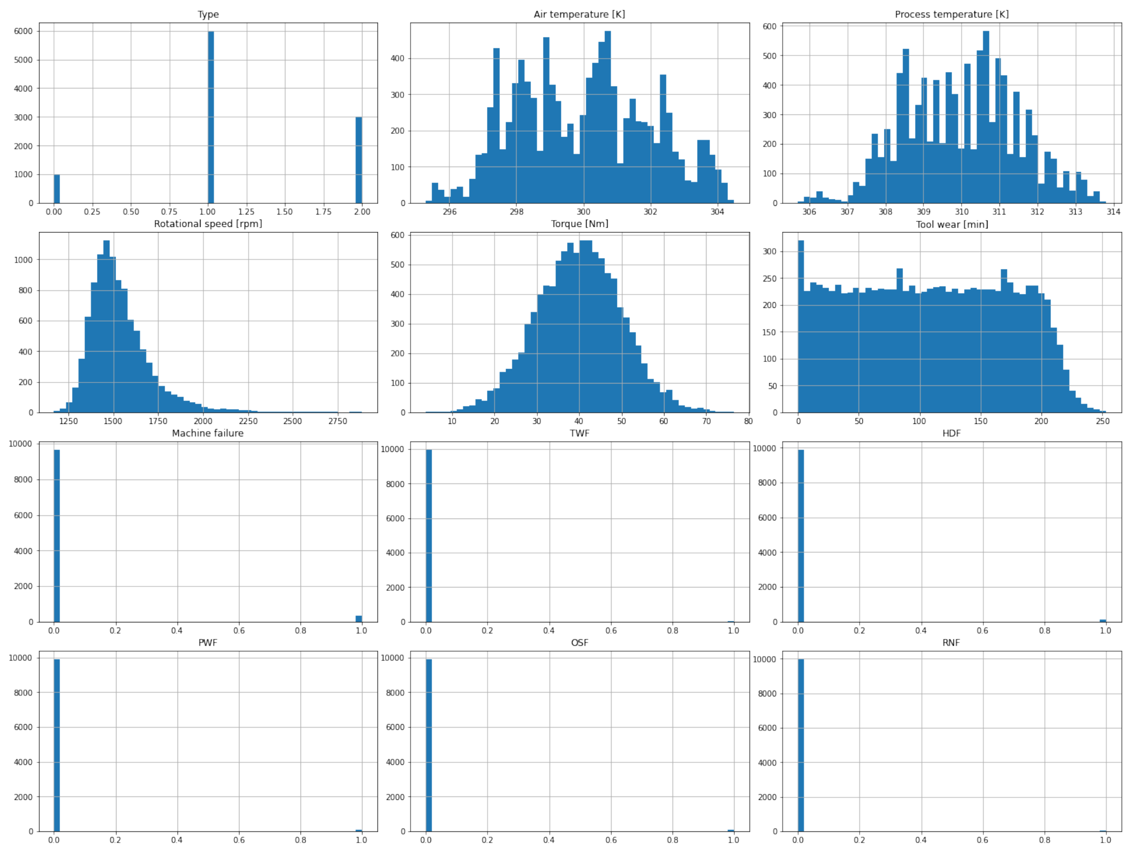
mitosheet visualization code

[22]:
```
# exploring data
```

[23]:
```
df.hist(bins=50, figsize=(20,15))
plt.tight_layout(pad=0.4)
plt.show()
```

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

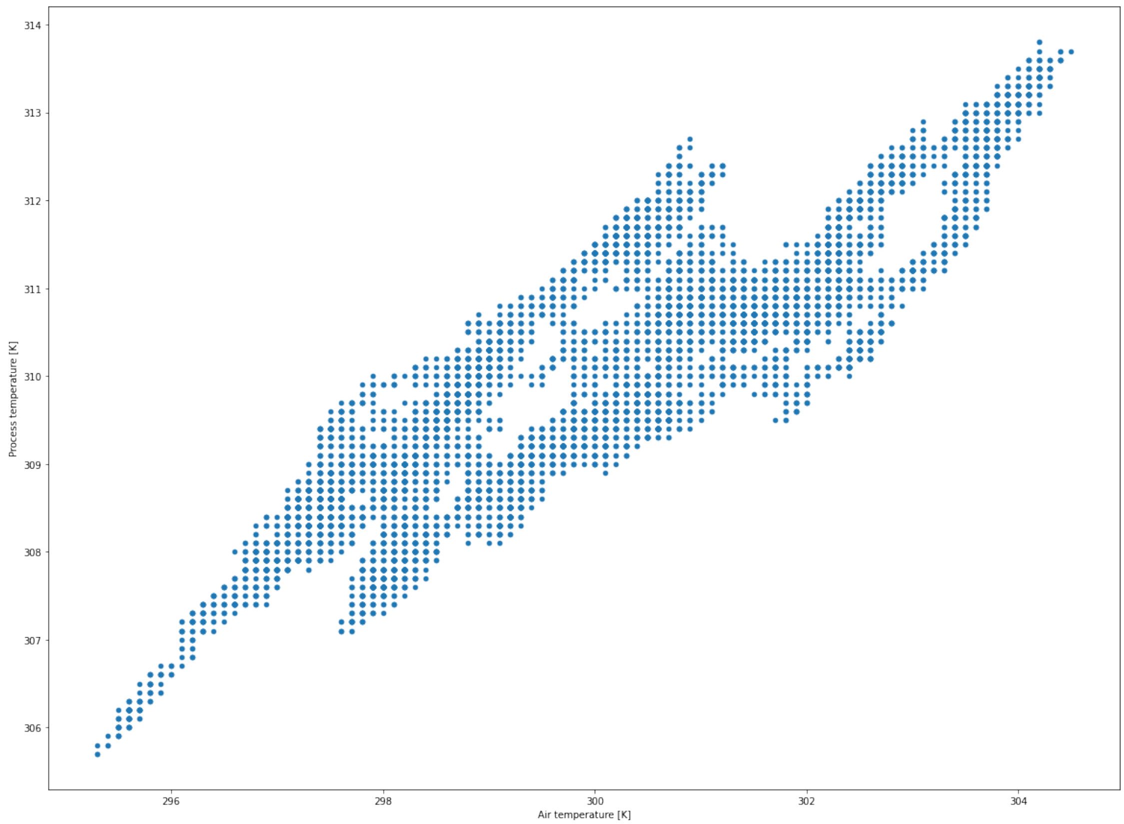


[24]:
```
df.plot.scatter(y = 'Type',x='Air temperature [K]', figsize=(20,15))
plt.show()
```

<IPython.core.display.Javascript object>

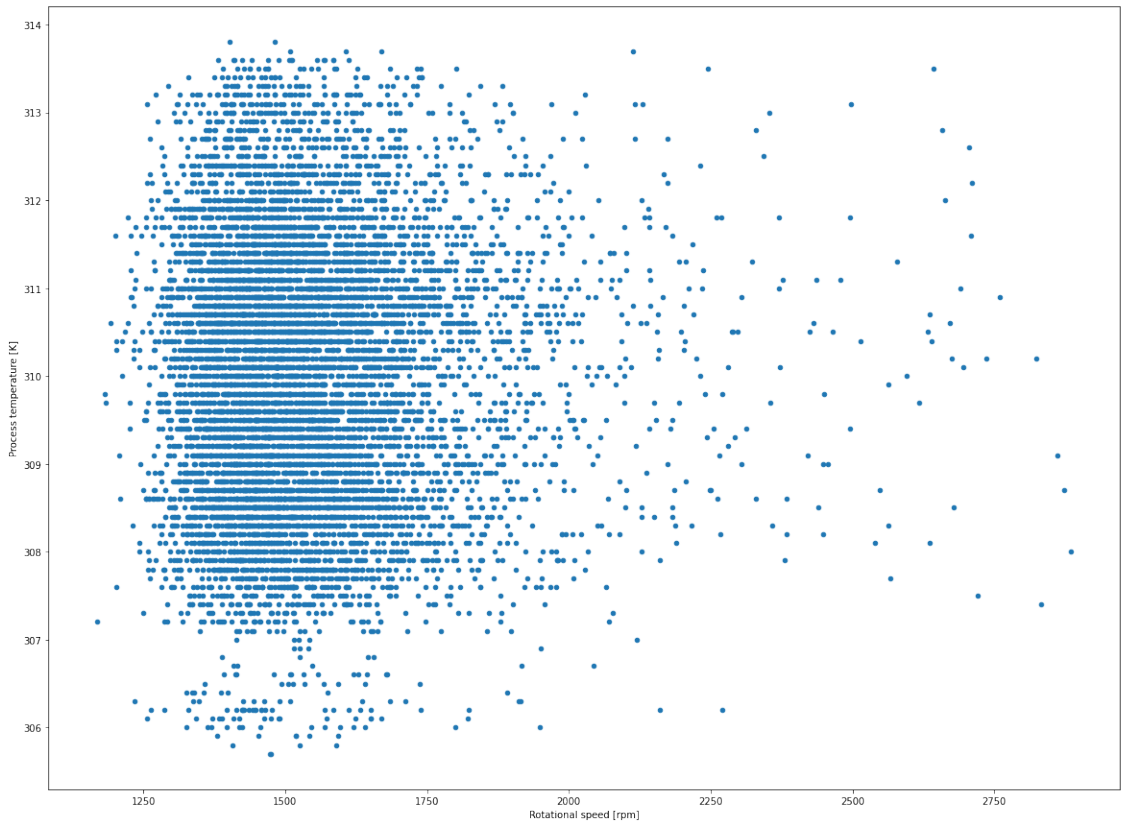
[25]:
```
df.plot.scatter(y = 'Process temperature [K]',x='Air temperature [K]',␣
↪figsize=(20,15))
plt.show()
```

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



[26]:
```
df.plot.scatter(y = 'Process temperature [K]',x='Rotational speed [rpm]',␣
 ↪figsize=(20,15))
plt.show()
```

<IPython.core.display.Javascript object>



[27]:
```python
#confusion matrix
```

[28]:
```python
import seaborn as sns

#correlation matrix
numeric_col = ['Type', 'Air temperature [K]', 'Process temperature [K]',
       'Rotational speed [rpm]', 'Torque [Nm]', 'Tool wear [min]',
        'TWF', 'HDF', 'PWF', 'OSF', 'RNF','Machine failure']
# Correlation Matrix formation
corr_matrix = df.loc[:,numeric_col].corr()
print(corr_matrix)
#Using heatmap to visualize the correlation matrix
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(corr_matrix, annot=True,ax=ax)
```

```
                             Type  Air temperature [K]  \
Type                     1.000000             0.017599
Air temperature [K]      0.017599             1.000000
Process temperature [K]  0.013444             0.876107
Rotational speed [rpm]  -0.002693             0.022670
Torque [Nm]              0.004011            -0.013778
```

Tool wear [min]	-0.003930	0.013853
TWF	-0.005349	0.009955
HDF	0.000108	0.137831
PWF	0.012121	0.003470
OSF	-0.021211	0.001988
RNF	-0.022147	0.017688
Machine failure	-0.005152	0.082556

	Process temperature [K]	Rotational speed [rpm]
Type	0.013444	-0.002693
Air temperature [K]	0.876107	0.022670
Process temperature [K]	1.000000	0.019277
Rotational speed [rpm]	0.019277	1.000000
Torque [Nm]	-0.014061	-0.875027
Tool wear [min]	0.013488	0.000223
TWF	0.007315	0.010389
HDF	0.056933	-0.121241
PWF	-0.003355	0.123018
OSF	0.004554	-0.104575
RNF	0.022279	-0.013088
Machine failure	0.035946	-0.044188

	Torque [Nm]	Tool wear [min]	TWF	HDF
Type	0.004011	-0.003930	-0.005349	0.000108
Air temperature [K]	-0.013778	0.013853	0.009955	0.137831
Process temperature [K]	-0.014061	0.013488	0.007315	0.056933
Rotational speed [rpm]	-0.875027	0.000223	0.010389	-0.121241
Torque [Nm]	1.000000	-0.003093	-0.014662	0.142610
Tool wear [min]	-0.003093	1.000000	0.115792	-0.001287
TWF	-0.014662	0.115792	1.000000	-0.007332
HDF	0.142610	-0.001287	-0.007332	1.000000
PWF	0.083781	-0.009334	0.008577	0.018443
OSF	0.183465	0.155894	0.038243	0.046396
RNF	0.016136	0.011326	0.030970	-0.004706
Machine failure	0.191321	0.105448	0.362904	0.575800

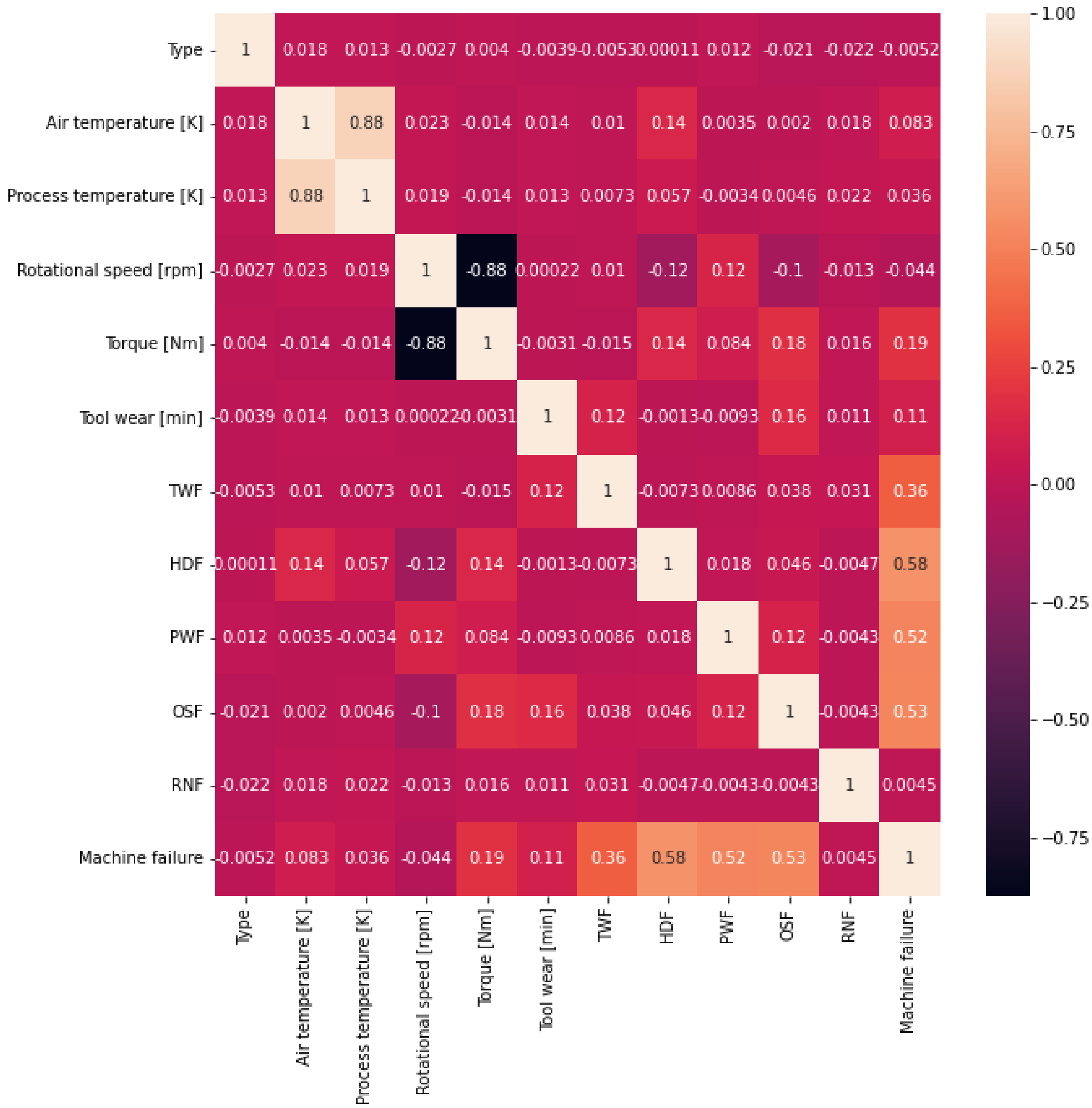
	PWF	OSF	RNF	Machine failure
Type	0.012121	-0.021211	-0.022147	-0.005152
Air temperature [K]	0.003470	0.001988	0.017688	0.082556
Process temperature [K]	-0.003355	0.004554	0.022279	0.035946
Rotational speed [rpm]	0.123018	-0.104575	-0.013088	-0.044188
Torque [Nm]	0.083781	0.183465	0.016136	0.191321
Tool wear [min]	-0.009334	0.155894	0.011326	0.105448
TWF	0.008577	0.038243	0.030970	0.362904
HDF	0.018443	0.046396	-0.004706	0.575800
PWF	1.000000	0.115836	-0.004273	0.522812
OSF	0.115836	1.000000	-0.004341	0.531083
RNF	-0.004273	-0.004341	1.000000	0.004516

```
Machine failure         0.522812  0.531083  0.004516         1.000000
```

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

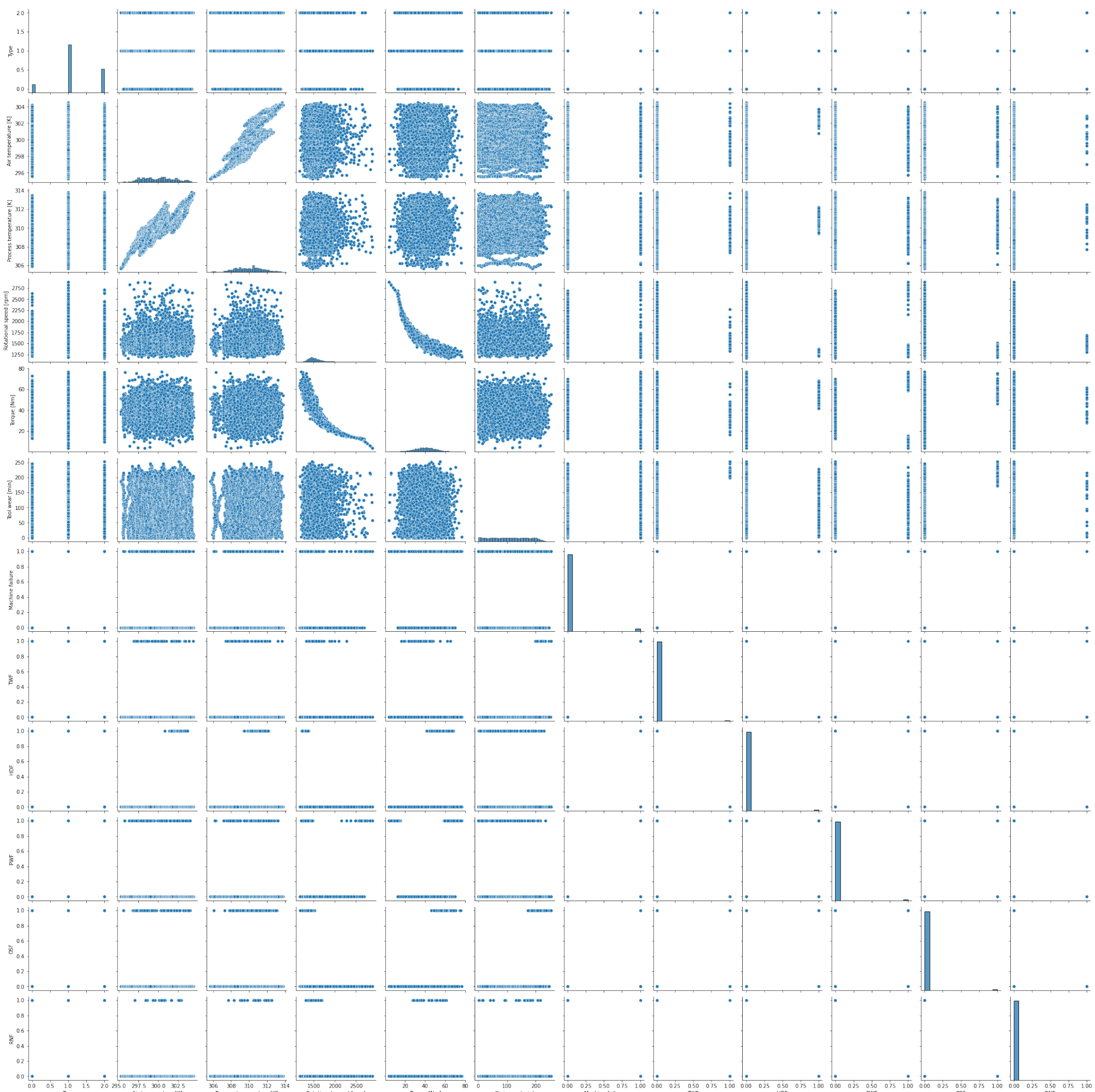
[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd31c989890>
```

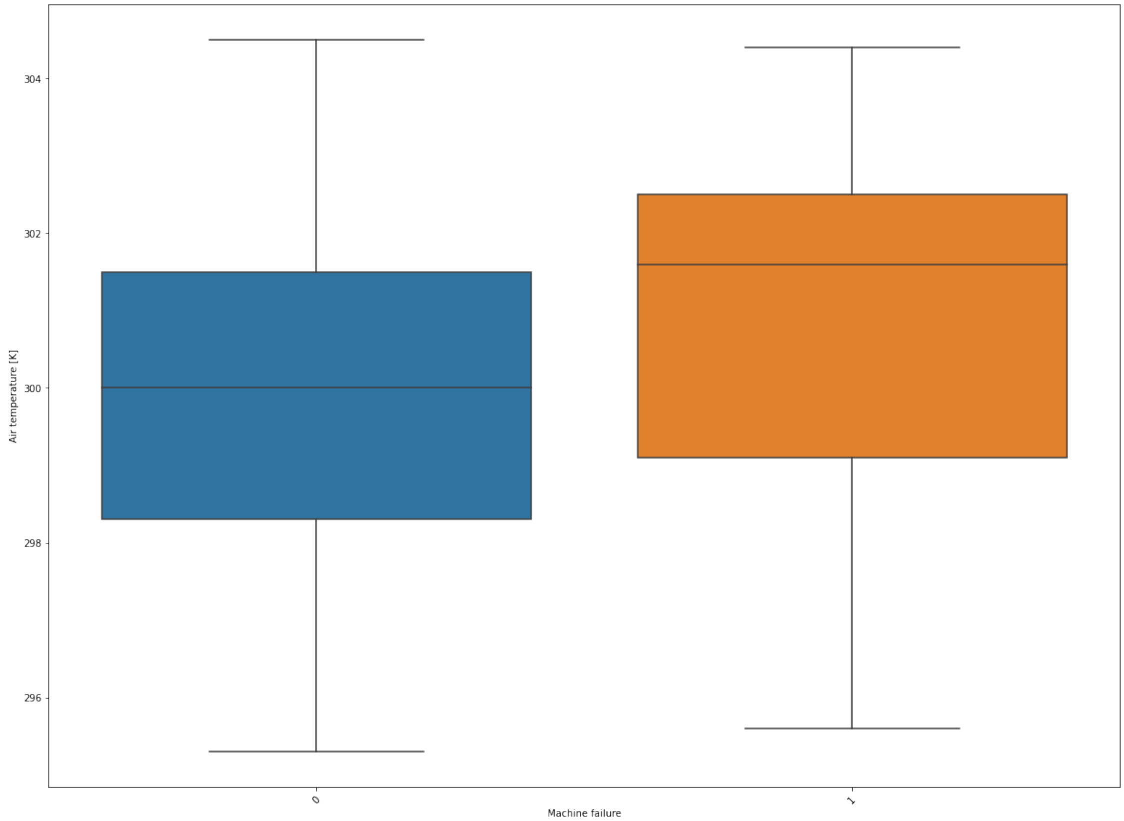


[29]:

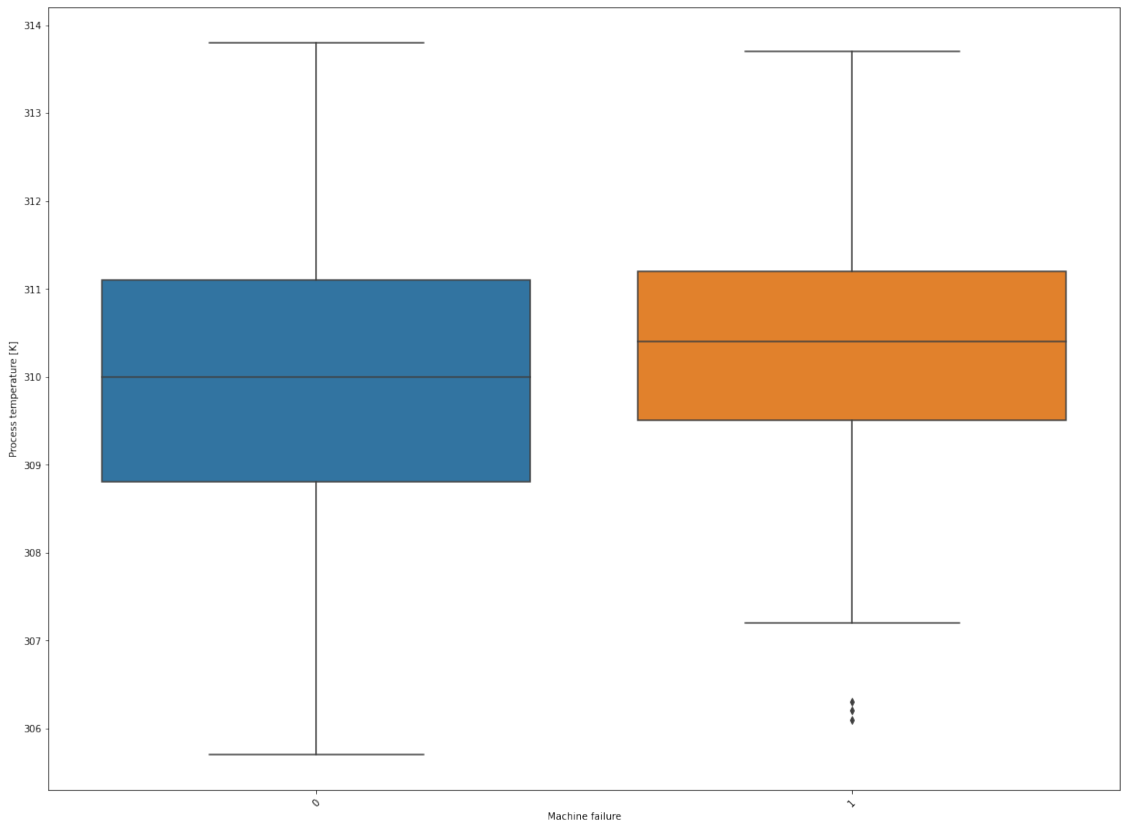
```python
import matplotlib.pyplot as plt
import seaborn as sns
sns.pairplot(df, kind="scatter")
plt.show()
```



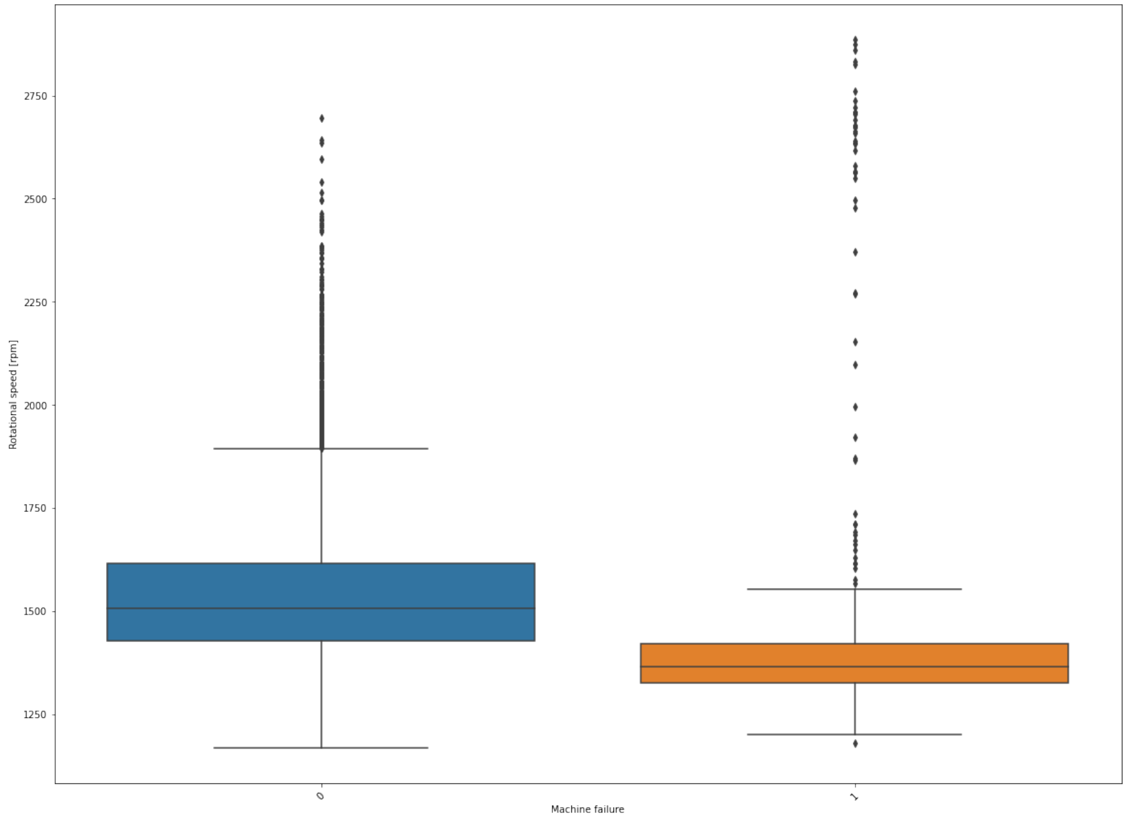
[30]:
```
plt.figure(figsize=(20,15))
plt.xticks(rotation=45)
sns.boxplot(data = df, y = 'Air temperature [K]', x = 'Machine failure');
```



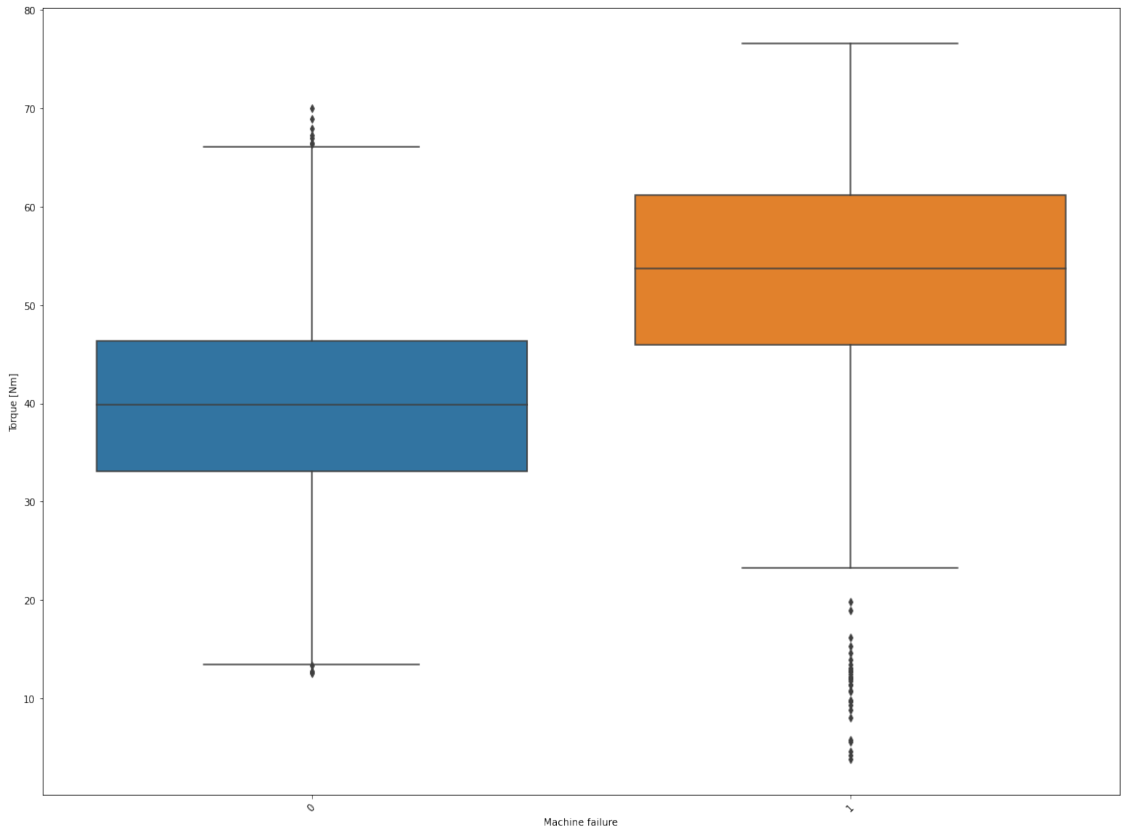
[31]:
```
plt.figure(figsize=(20,15))
plt.xticks(rotation=45)
sns.boxplot(data = df, y = 'Process temperature [K]', x = 'Machine failure');
```



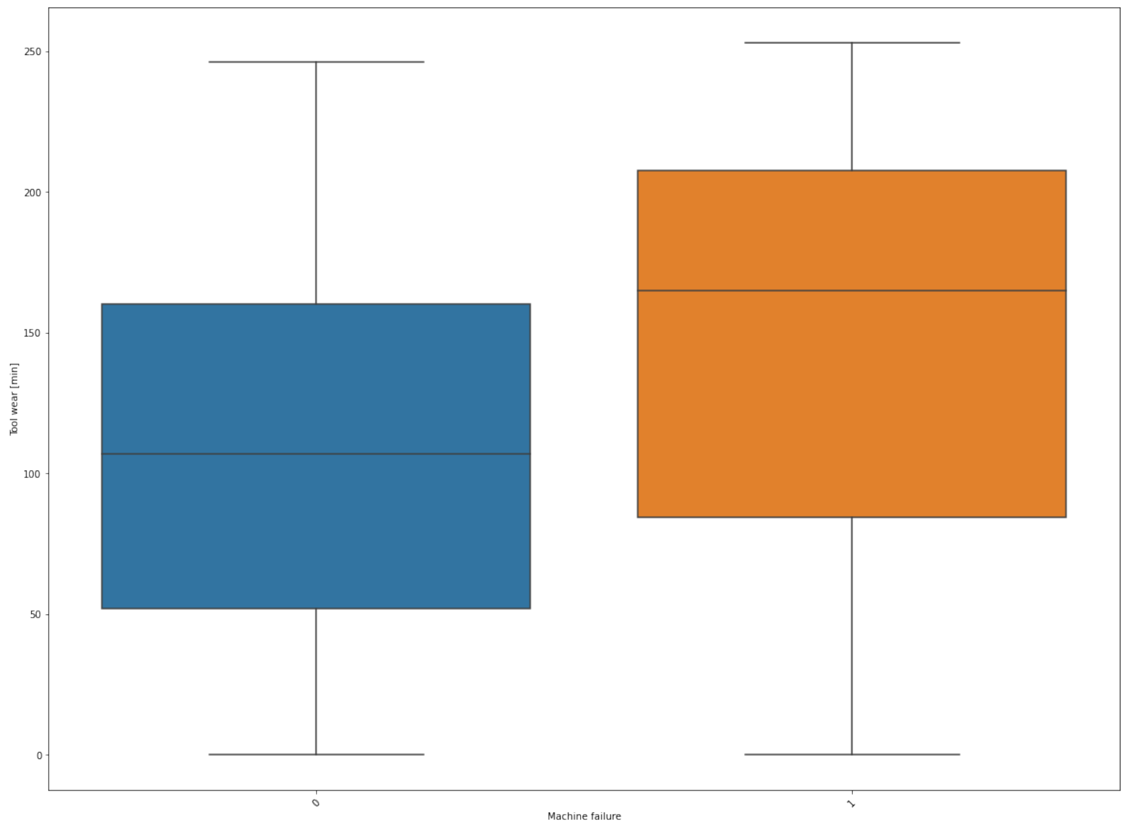
[32]:
```
plt.figure(figsize=(20,15))
plt.xticks(rotation=45)
sns.boxplot(data = df, y = 'Rotational speed [rpm]', x = 'Machine failure');
```



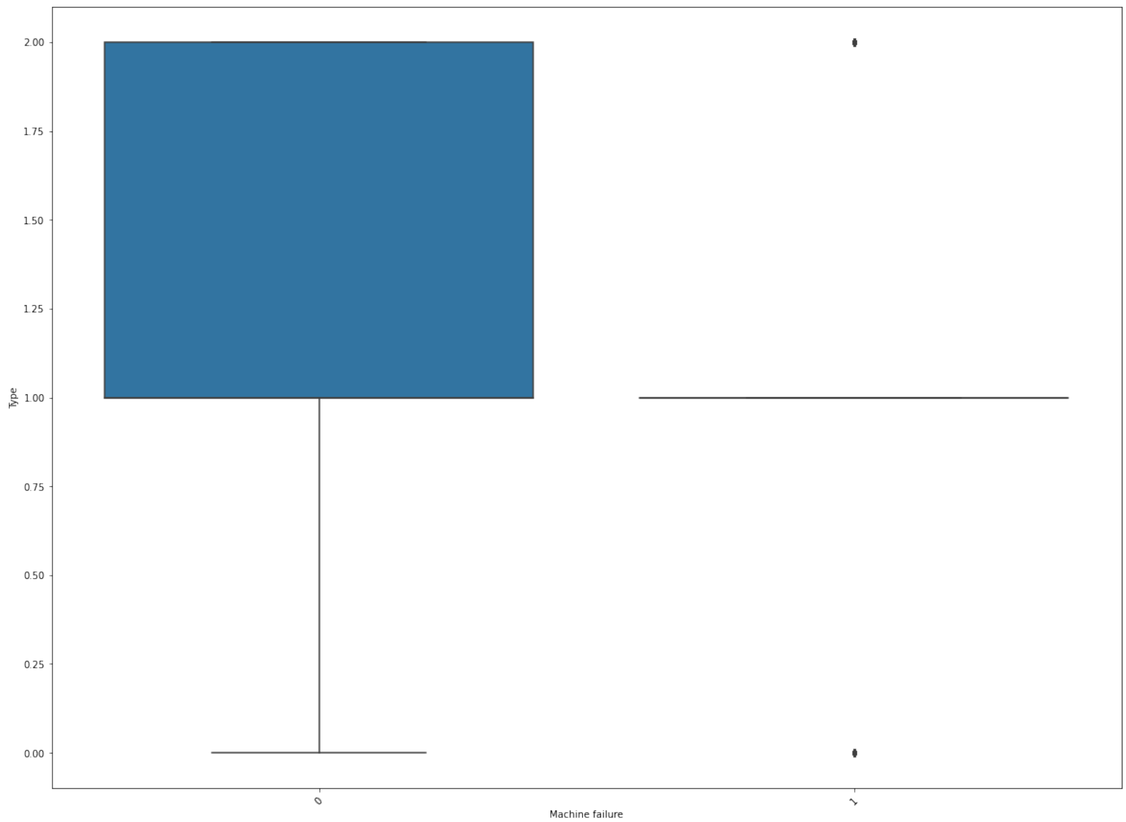
[33]:
```
plt.figure(figsize=(20,15))
plt.xticks(rotation=45)
sns.boxplot(data = df, y = 'Torque [Nm]', x = 'Machine failure');
```



[34]:
```
plt.figure(figsize=(20,15))
plt.xticks(rotation=45)
sns.boxplot(data = df, y = 'Tool wear [min]', x = 'Machine failure');
```



[35]:
```
plt.figure(figsize=(20,15))
plt.xticks(rotation=45)
sns.boxplot(data = df, y = 'Type', x = 'Machine failure');
```



0.0.2 BNN

[36]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow_datasets as tfds
import tensorflow_probability as tfp
```

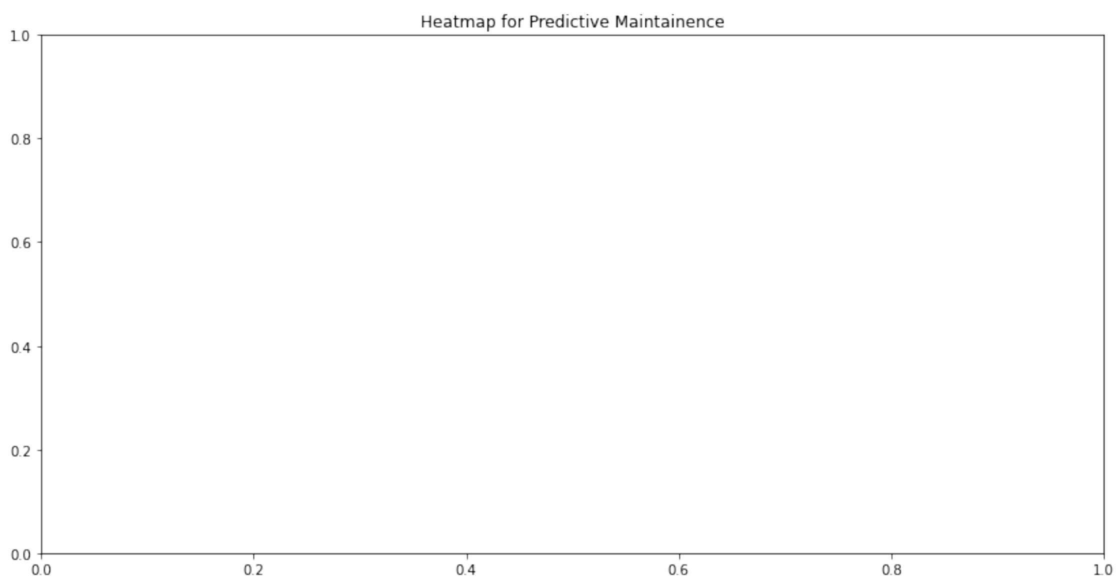
visualizing data

[37]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

plt.figure(figsize=(16,6))
plt.title("Simple Line Plot on Whole Data")
sns.lineplot(data=df)
```

[37]: `<matplotlib.axes._subplots.AxesSubplot at 0x7fd31c442fd0>`

[38]:
```
#heatmaps on whole data
plt.figure(figsize=(14,7))
# Add title
plt.title("Heatmap for Predictive Maintainence")
# Heatmap
#sns.heatmap(data=df['Machine failure'], annot=True)
# Add label for horizontal axis
#plt.xlabel("Axis")
```

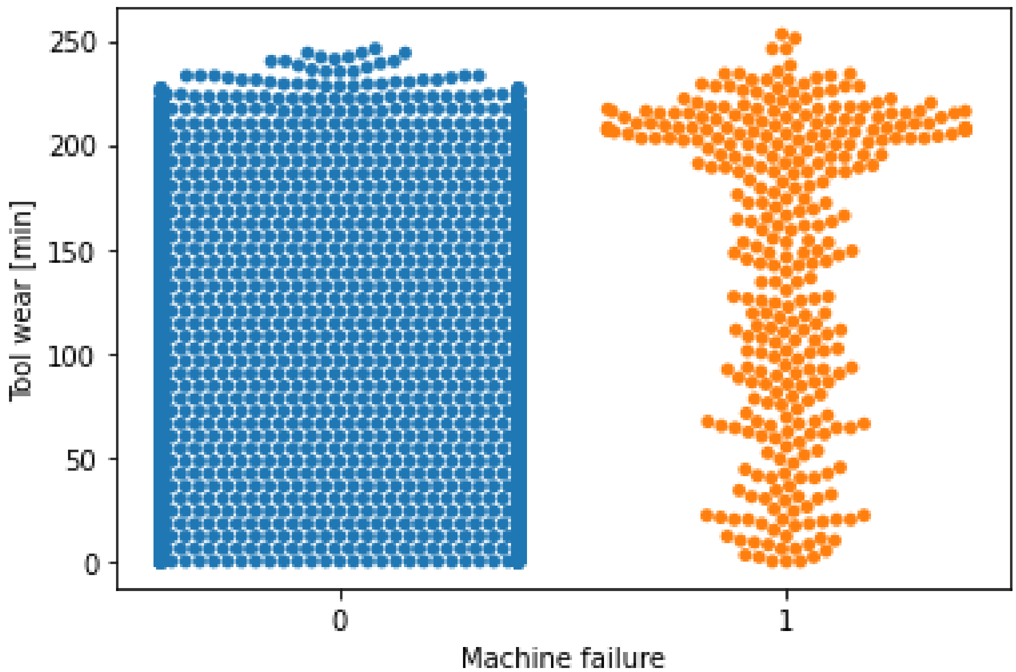
[38]: Text(0.5, 1.0, 'Heatmap for Predictive Maintainence')



[39]:
```
sns.swarmplot(x=df['Machine failure'],y=df['Tool wear [min]'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning:
89.6% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

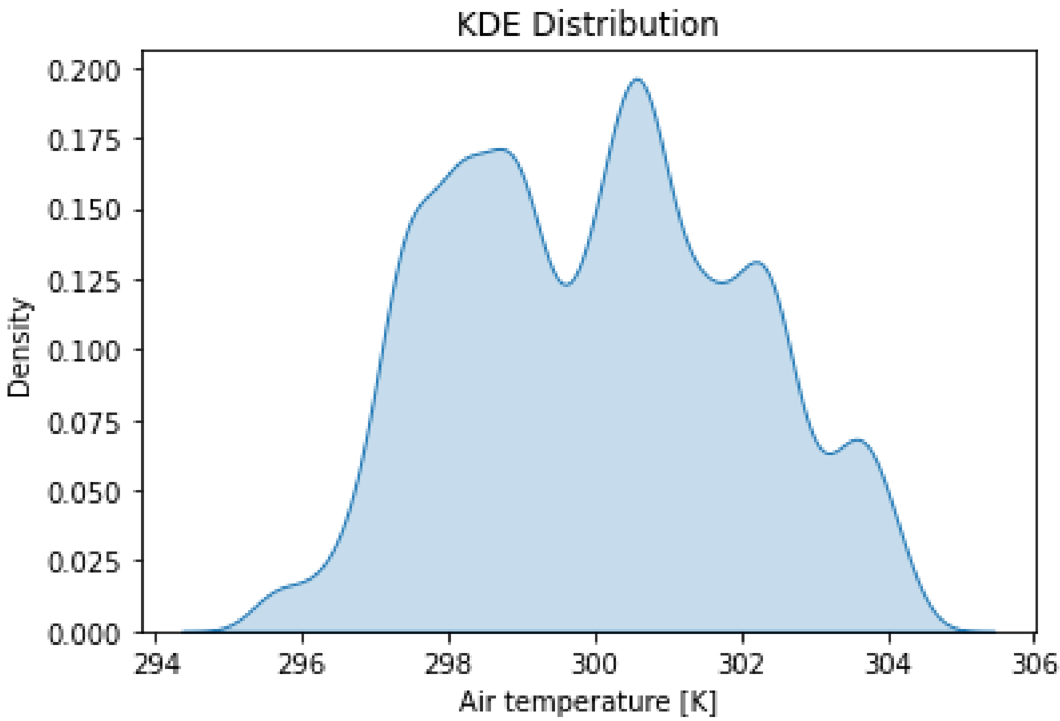
[39]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd2b1635250>
```



[40]:
```python
#stripplot
```

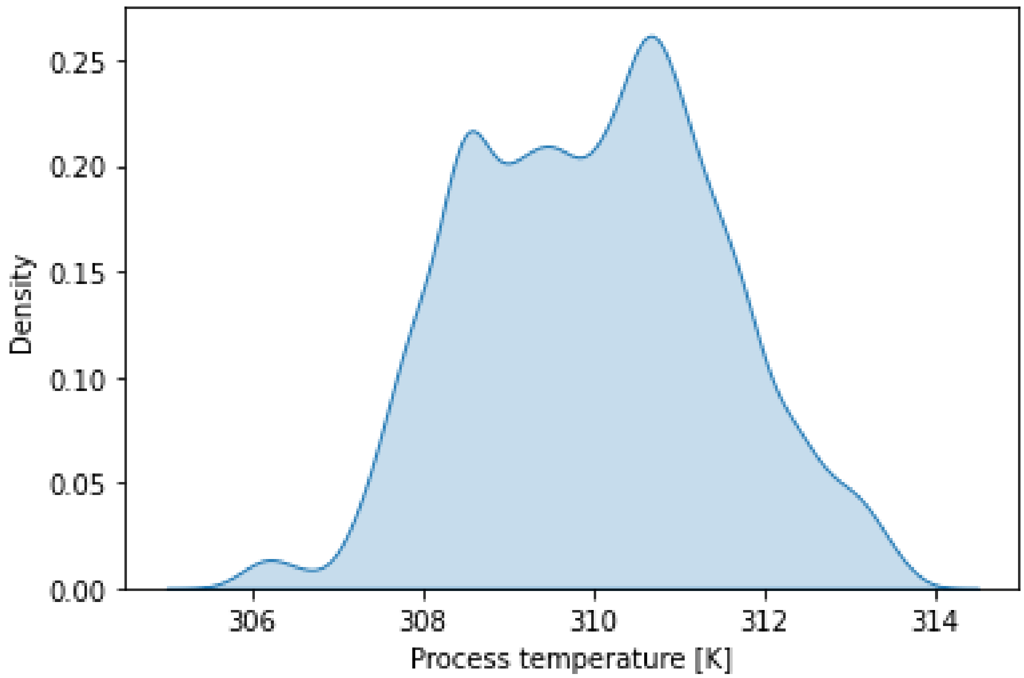
[41]:
```python
#distribution
#for i in df:
sns.kdeplot(data=df['Air temperature [K]'], label='Air temperature [K]',␣
 ↪shade=True)

plt.title('KDE Distribution')
```

[41]:
```
Text(0.5, 1.0, 'KDE Distribution')
```



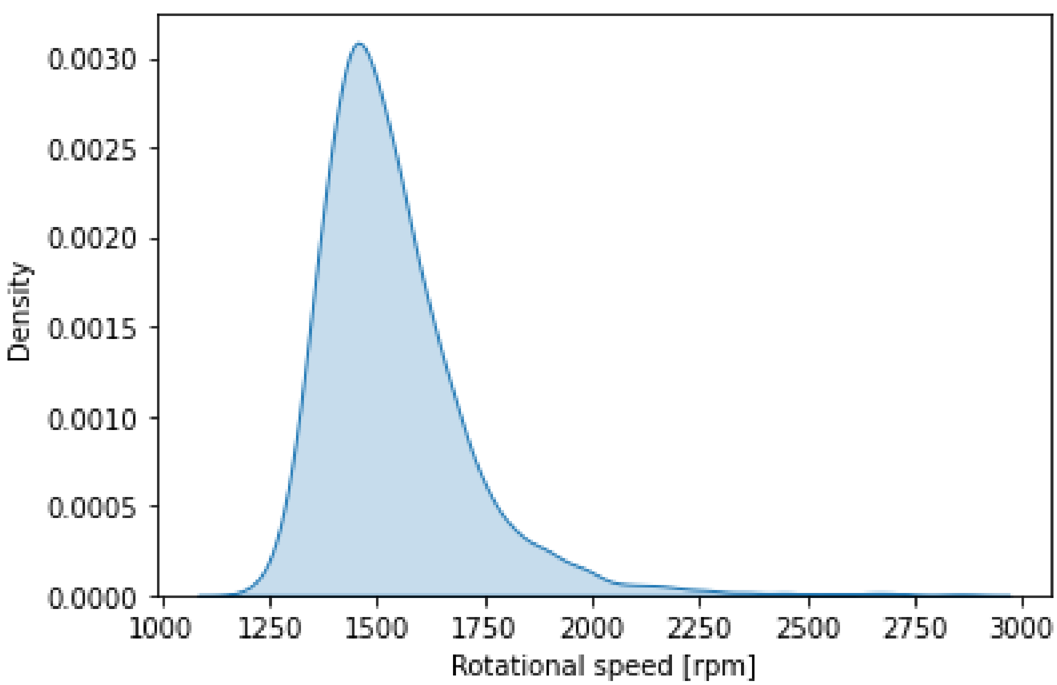
[42]:
```
sns.kdeplot(data=df['Process temperature [K]'], label='Process temperature␣
↪[K]', shade=True)
```

[42]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd2b1534110>



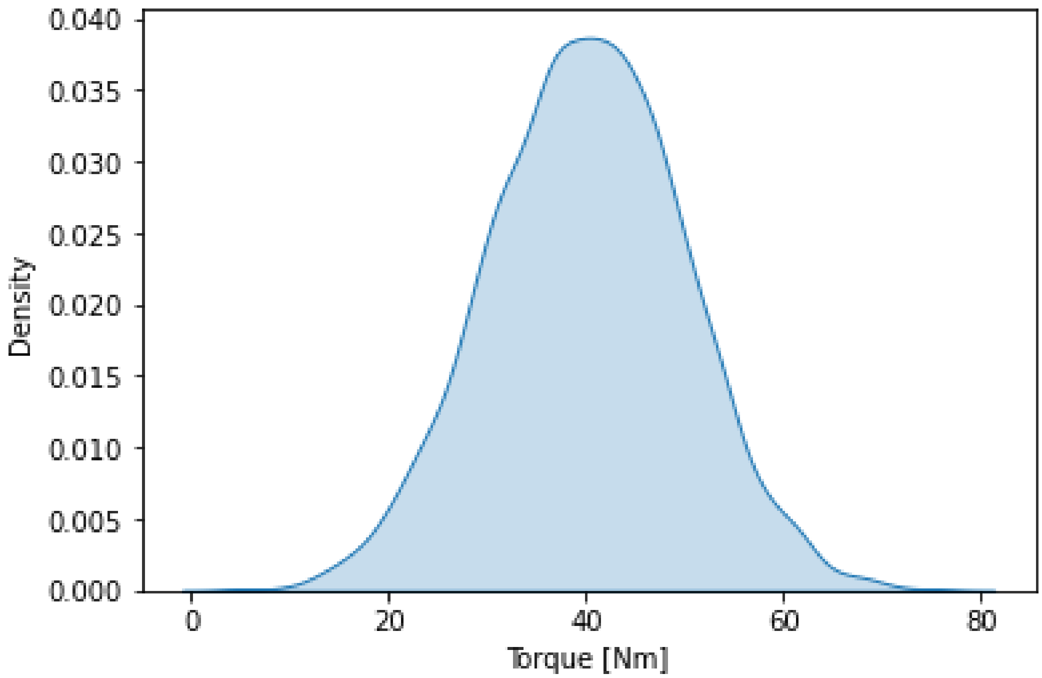
[43]:
```
sns.kdeplot(data=df['Rotational speed [rpm]'], label='Rotational speed [rpm]',␣
↪shade=True)
```

[43]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd2b14e8910>
```



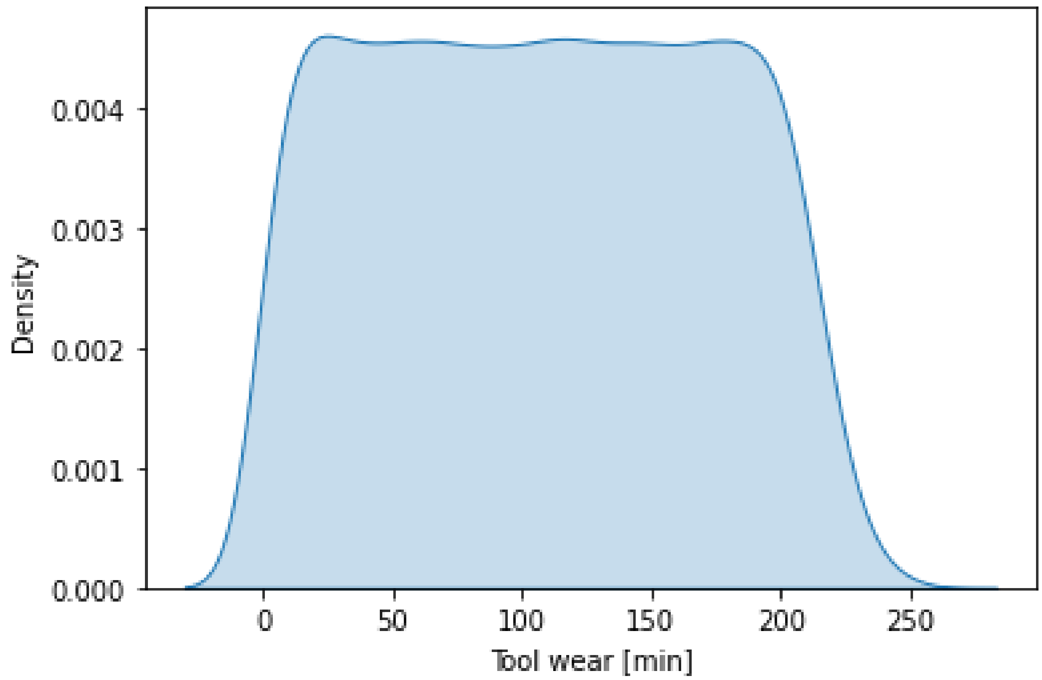
[44]:
```
sns.kdeplot(data=df['Torque [Nm]'], label='Torque [Nm]', shade=True)
```

[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd2b14960d0>



[45]:
```
sns.kdeplot(data=df['Tool wear [min]'], label='Tool wear [min]', shade=True)
```

[45]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd2b1406750>



Create training and evaluation datasets

[46]:
```
df.columns
```

[46]:
```
Index(['Type', 'Air temperature [K]', 'Process temperature [K]',
       'Rotational speed [rpm]', 'Torque [Nm]', 'Tool wear [min]',
       'Machine failure', 'TWF', 'HDF', 'PWF', 'OSF', 'RNF'],
      dtype='object')
```

[47]:
```
from sklearn.model_selection import train_test_split

#first moving target variable "Machine Failure" to end and then defining X and y
df = df[['Type', 'Air temperature [K]', 'Process temperature [K]',
       'Rotational speed [rpm]', 'Torque [Nm]', 'Tool wear [min]',
        'TWF', 'HDF', 'PWF', 'OSF', 'RNF','Machine failure']]
print(df.shape)
# excluding last variable for target variable
X = df.iloc[:, :-1]
print(X.shape)
# making last variable as target variable
y = df.iloc[:, -1]
print(y.shape)
# using 70:30 split for making training and testing datasets and using random␣
↪state as 42 to repeat this random split.
```

```
X_train,X_test,y_train,y_test = train_test_split(X, y,test_size=0.
↪3,random_state=42)
```

```
(10000, 12)
(10000, 11)
(10000,)
```

[48]:
```
# the shapes of X_train,X_test,y_train,y_test
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(7000, 11)
(3000, 11)
(7000,)
(3000,)
```

[49]:
```
print(X_train.shape)
print(y_train.shape)
```

```
(7000, 11)
(7000,)
```

[50]:
```
y_train.head()
```

[50]:
```
9069    0
2603    0
7738    0
1579    0
5058    0
Name: Machine failure, dtype: int64
```

[51]:
```
# correct
#done
#train dataset
train_d = pd.DataFrame(X_train)
train_d['y_train'] = y_train
print(train_d.shape)
print(train_d.shape)

#test dataset
test_d = pd.DataFrame(X_test)
test_d['y_test'] = y_test
print(test_d.shape)
print(test_d.shape)
```

```
(7000, 12)
```

```
(7000, 12)
(3000, 12)
(3000, 12)
```

[52]:
```
train_d.head()
```

[52]:
```
      Type  Air temperature [K]  Process temperature [K]  \
9069   2.0                297.2                    308.2
2603   2.0                299.3                    309.2
7738   2.0                300.5                    312.0
1579   1.0                298.3                    308.3
5058   1.0                303.9                    312.9

      Rotational speed [rpm]  Torque [Nm]  Tool wear [min]  TWF  HDF  PWF  \
9069                    1678         28.1              133    0    0    0
2603                    1334         46.3               31    0    0    0
7738                    1263         60.8              146    0    0    0
1579                    1444         43.8              176    0    0    0
5058                    1526         42.5              194    0    0    0

      OSF  RNF  y_train
9069    0    0        0
2603    0    0        0
7738    0    0        0
1579    0    0        0
5058    0    0        0
```

[53]:
```
test_d.head()
```

[53]:
```
      Type  Air temperature [K]  Process temperature [K]  \
6252   1.0                300.8                    310.3
4684   2.0                303.6                    311.8
1731   2.0                298.3                    307.9
4742   1.0                303.3                    311.3
4521   1.0                302.4                    310.4

      Rotational speed [rpm]  Torque [Nm]  Tool wear [min]  TWF  HDF  PWF  \
6252                    1538         36.1              198    0    0    0
4684                    1421         44.8              101    0    0    0
1731                    1485         42.0              117    0    0    0
4742                    1592         33.7               14    0    0    0
4521                    1865         23.9              129    0    0    0

      OSF  RNF  y_test
6252    0    0       0
4684    0    0       1
1731    0    0       0
```

```
4742    0    0        0
4521    0    0        0
```

Compile, train, and evaluate the model

[54]:
```
# from here will write in the form of functions
```

Create model inputs

Experiment 1: standard neural network(Non-bayesian neural network)

[55]:
```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from keras.models import Sequential # to initialize NN
from keras.layers import Dense # to build layers

classifier = Sequential()
classifier.add(Dense(units = 5, input_dim = X_train.shape[1])) # changed this
classifier.add(Dense(units = 3, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =␣
↪['accuracy'])
history = classifier.fit(X_train, y_train, epochs=50)
#validation_data = (np.asarray(X_test), np.asarray(y_test)),verbose=0
test_loss, test_acc = classifier.evaluate(X_test,  y_test, verbose=2)
print('\nTest accuracy:', test_acc)
print('\nTest loss:', test_loss)
```

```
Epoch 1/50
219/219 [==============================] - 1s 2ms/step - loss: 0.3808 -
accuracy: 0.9453
Epoch 2/50
219/219 [==============================] - 1s 3ms/step - loss: 0.1759 -
accuracy: 0.9649
Epoch 3/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1574 -
accuracy: 0.9649
Epoch 4/50
219/219 [==============================] - 1s 3ms/step - loss: 0.1505 -
accuracy: 0.9649
Epoch 5/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1466 -
accuracy: 0.9649
Epoch 6/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1451 -
accuracy: 0.9649
```

```
Epoch 7/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1449 -
accuracy: 0.9649
Epoch 8/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1416 -
accuracy: 0.9649
Epoch 9/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1406 -
accuracy: 0.9649
Epoch 10/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1373 -
accuracy: 0.9649
Epoch 11/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1349 -
accuracy: 0.9649
Epoch 12/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1340 -
accuracy: 0.9649
Epoch 13/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1299 -
accuracy: 0.9649
Epoch 14/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1294 -
accuracy: 0.9649
Epoch 15/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1246 -
accuracy: 0.9649
Epoch 16/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1156 -
accuracy: 0.9649
Epoch 17/50
219/219 [==============================] - 0s 2ms/step - loss: 0.1038 -
accuracy: 0.9649
Epoch 18/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0979 -
accuracy: 0.9649
Epoch 19/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0810 -
accuracy: 0.9649
Epoch 20/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0720 -
accuracy: 0.9649
Epoch 21/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0659 -
accuracy: 0.9649
Epoch 22/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0483 -
accuracy: 0.9744
```

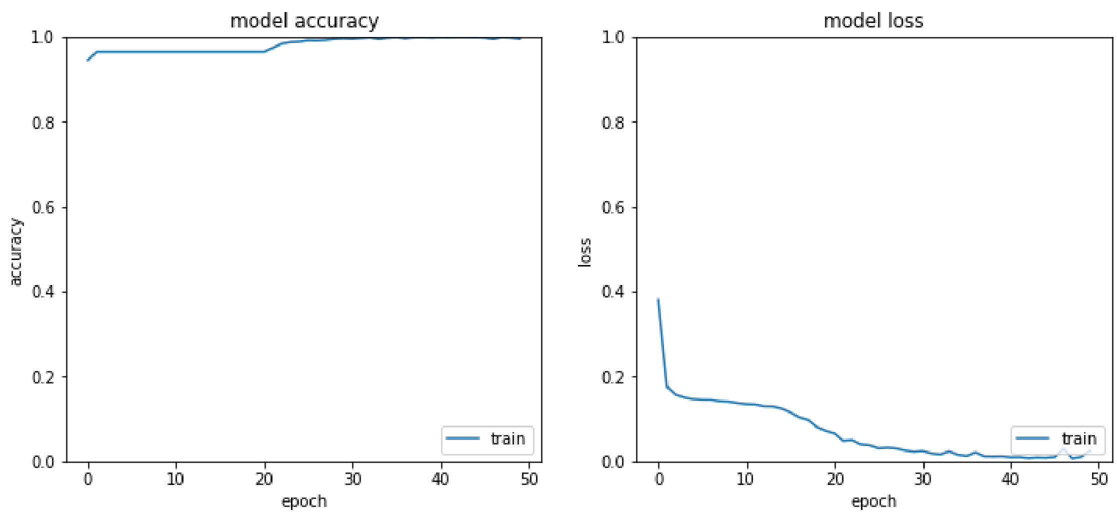
```
Epoch 23/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0502 -
accuracy: 0.9843
Epoch 24/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0398 -
accuracy: 0.9880
Epoch 25/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0381 -
accuracy: 0.9890
Epoch 26/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0308 -
accuracy: 0.9921
Epoch 27/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0323 -
accuracy: 0.9916
Epoch 28/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0307 -
accuracy: 0.9929
Epoch 29/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0257 -
accuracy: 0.9956
Epoch 30/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0227 -
accuracy: 0.9964
Epoch 31/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0244 -
accuracy: 0.9957
Epoch 32/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0187 -
accuracy: 0.9971
Epoch 33/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0161 -
accuracy: 0.9986
Epoch 34/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0234 -
accuracy: 0.9950
Epoch 35/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0155 -
accuracy: 0.9979
Epoch 36/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0125 -
accuracy: 0.9991
Epoch 37/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0208 -
accuracy: 0.9966
Epoch 38/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0118 -
accuracy: 0.9989
```

```
Epoch 39/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0114 -
accuracy: 0.9990
Epoch 40/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0118 -
accuracy: 0.9981
Epoch 41/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0097 -
accuracy: 0.9990
Epoch 42/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0099 -
accuracy: 0.9986
Epoch 43/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0075 -
accuracy: 0.9994
Epoch 44/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0089 -
accuracy: 0.9990
Epoch 45/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0083 -
accuracy: 0.9990
Epoch 46/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0103 -
accuracy: 0.9981
Epoch 47/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0295 -
accuracy: 0.9951
Epoch 48/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0065 -
accuracy: 0.9993
Epoch 49/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0107 -
accuracy: 0.9980
Epoch 50/50
219/219 [==============================] - 0s 2ms/step - loss: 0.0248 -
accuracy: 0.9956
94/94 - 0s - loss: 0.0047 - accuracy: 1.0000 - 272ms/epoch - 3ms/step

Test accuracy: 1.0

Test loss: 0.004710679408162832
```

train accuracy: 0.9649, loss: 0.1522 after 50 epochs test accuracy: 0.9690, loss: 0.1385

[56]:
```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'])
#plt.plot(history.history['val_accuracy'])
```

```
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='lower right')
plt.ylim(0, 1)
# summarize history for loss
plt.subplot(1,2,2)
plt.plot(history.history['loss'])
#plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='lower right')
plt.ylim(0, 1)
plt.show()
```



[57]:
```
classifier.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 5)                 65

 dense_1 (Dense)             (None, 3)                 18

 dense_2 (Dense)             (None, 1)                 4

=================================================================
Total params: 87
```

```
Trainable params: 87
Non-trainable params: 0
_________________________________________________________________
```

```
[58]: # checking the probabilities
probability_model = Sequential([classifier, tf.keras.layers.Softmax()])
predictions = probability_model.predict(X_test)
predictions[0]
```

```
[58]: array([1.], dtype=float32)
```

```
[59]: np.argmax(predictions[0])
```

```
[59]: 0
```

```
[60]: y_test[0]
```

```
[60]: 0
```

```
[61]: predictions
```

```
[61]: array([[1.],
       [1.],
       [1.],
       …,
       [1.],
       [1.],
       [1.]], dtype=float32)
```

```
[62]: y_test.nunique
```

```
[62]: <bound method IndexOpsMixin.nunique of 6252    0
4684    1
1731    0
4742    0
4521    0
       ..
8014    0
1074    0
3063    0
6487    0
4705    0
Name: Machine failure, Length: 3000, dtype: int64>
```

Experiment 2: Bayesian neural network (BNN)

dependencies and prerequisites

[63]:
```
from pprint import pprint
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

import tensorflow.compat.v2 as tf
tf.enable_v2_behavior()

import tensorflow_probability as tfp

sns.reset_defaults()
sns.set_context(context='talk',font_scale=0.7)
plt.rcParams['image.cmap'] = 'viridis'

%matplotlib inline

tfd = tfp.distributions
tfb = tfp.bijectors
```

define priors and other functions

[64]:
```
# to build the bnn
```

define bnn functions and class

[65]:
```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from keras.models import Sequential # to initialize NN
from keras.layers import Dense # to build layers
'''
classifier = Sequential()
classifier.add(Dense(units = 8, input_dim = X_train.shape[1])) # changed this
classifier.add(Dense(units = 4, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =␣
↪['accuracy'])
classifier.fit(X_train, y_train, epochs=100)
test_loss, test_acc = classifier.evaluate(X_test,  y_test, verbose=2)
print('\nTest accuracy:', test_acc)

'''
```

[65]:
```
"\nclassifier = Sequential()\nclassifier.add(Dense(units = 8, input_dim =
X_train.shape[1])) # changed this\nclassifier.add(Dense(units = 4, activation =
'relu'))\nclassifier.add(Dense(units = 1, activation =
'sigmoid'))\nclassifier.compile(optimizer = 'adam', loss =
```

```
'binary_crossentropy', metrics = ['accuracy'])\nclassifier.fit(X_train, y_train,
epochs=100)\ntest_loss, test_acc = classifier.evaluate(X_test,  y_test,
verbose=2)\nprint('\nTest accuracy:', test_acc)\n\n"
```

target is machine failure variable

[66]:
```
from sklearn.model_selection import train_test_split

#first moving target variable "Machine Failure" to end and then defining X and y
df = df[['Type', 'Air temperature [K]', 'Process temperature [K]',
       'Rotational speed [rpm]', 'Torque [Nm]', 'Tool wear [min]',
        'TWF', 'HDF', 'PWF', 'OSF', 'RNF','Machine failure']]
print(df.shape)
# excluding last variable for target variable
X = df.iloc[:, :-1]
print(X.shape)
# making last variable as target variable
y = df.iloc[:, -1]
print(y.shape)
# using 70:30 split for making training and testing datasets and using random␣
↪state as 42 to repeat this random split.
X_train,X_test,y_train,y_test = train_test_split(X, y,test_size=0.
↪3,random_state=42)
```

```
(10000, 12)
(10000, 11)
(10000,)
```

[67]:
```
dist = tfp.distributions
dataset_size = len(X_train)
kl_divergence_function = (lambda q, p, _: dist.kl_divergence(q, p) / tf.
↪cast(dataset_size, dtype=tf.float32))

model_tfp = tf.keras.Sequential([
    tf.keras.Input(X_train.shape[1]),
    tfp.layers.DenseFlipout(16, kernel_divergence_fn=kl_divergence_function),#,␣
↪activation=tf.nn.relu),
    tfp.layers.DenseFlipout(6, kernel_divergence_fn=kl_divergence_function,␣
↪activation=tf.nn.relu ),
    tfp.layers.DenseFlipout(3, kernel_divergence_fn=kl_divergence_function,␣
↪activation=tf.nn.softmax),
])

learning_rate = 0.001
model_tfp.compile(optimizer=tf.keras.optimizers.
↪Adam(learning_rate),loss='binary_crossentropy',metrics=['accuracy'])
```

```
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:102: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:112: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
```

[68]:
```
model_tfp.fit(X_train, y_train, epochs=50)
test_loss, test_acc = model_tfp.evaluate(X_test,  y_test)
print('\nTest accuracy:', test_acc)
print('\nTest loss:', test_loss)
```

```
Epoch 1/50
219/219 [==============================] - 3s 3ms/step - loss: 0.8826 -
accuracy: 0.1604
Epoch 2/50
219/219 [==============================] - 1s 3ms/step - loss: 0.5352 -
accuracy: 0.0457
Epoch 3/50
219/219 [==============================] - 1s 3ms/step - loss: 0.5301 -
accuracy: 0.0416
Epoch 4/50
219/219 [==============================] - 1s 3ms/step - loss: 0.5253 -
accuracy: 0.0384
Epoch 5/50
219/219 [==============================] - 1s 3ms/step - loss: 0.5218 -
accuracy: 0.0227
Epoch 6/50
219/219 [==============================] - 1s 3ms/step - loss: 0.5186 -
accuracy: 0.3166
Epoch 7/50
219/219 [==============================] - 1s 3ms/step - loss: 0.5154 -
accuracy: 0.0011
Epoch 8/50
219/219 [==============================] - 1s 3ms/step - loss: 0.5125 -
accuracy: 0.0137
Epoch 9/50
219/219 [==============================] - 1s 3ms/step - loss: 0.5093 -
accuracy: 0.0356
Epoch 10/50
219/219 [==============================] - 1s 3ms/step - loss: 0.5066 -
accuracy: 0.1624
Epoch 11/50
219/219 [==============================] - 1s 3ms/step - loss: 0.5038 -
```

```
accuracy: 0.0133
Epoch 12/50
219/219 [==============================] - 1s 3ms/step - loss: 0.5012 -
accuracy: 0.2573
Epoch 13/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4987 -
accuracy: 0.7191
Epoch 14/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4971 -
accuracy: 0.5563
Epoch 15/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4945 -
accuracy: 0.3813
Epoch 16/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4928 -
accuracy: 0.0229
Epoch 17/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4911 -
accuracy: 0.1279
Epoch 18/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4896 -
accuracy: 0.4983
Epoch 19/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4880 -
accuracy: 0.5884
Epoch 20/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4867 -
accuracy: 0.6009
Epoch 21/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4853 -
accuracy: 0.3571
Epoch 22/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4839 -
accuracy: 0.5247
Epoch 23/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4824 -
accuracy: 0.1337
Epoch 24/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4812 -
accuracy: 0.3340
Epoch 25/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4801 -
accuracy: 0.3097
Epoch 26/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4787 -
accuracy: 0.1249
Epoch 27/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4776 -
```

```
accuracy: 0.7490
Epoch 28/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4766 -
accuracy: 0.3694
Epoch 29/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4754 -
accuracy: 0.4960
Epoch 30/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4747 -
accuracy: 0.2961
Epoch 31/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4737 -
accuracy: 0.3577
Epoch 32/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4726 -
accuracy: 0.3296
Epoch 33/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4717 -
accuracy: 0.7987
Epoch 34/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4712 -
accuracy: 0.4844
Epoch 35/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4701 -
accuracy: 0.7370
Epoch 36/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4697 -
accuracy: 0.5600
Epoch 37/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4691 -
accuracy: 0.3601
Epoch 38/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4708 -
accuracy: 0.3040
Epoch 39/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4676 -
accuracy: 0.4377
Epoch 40/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4671 -
accuracy: 0.2989
Epoch 41/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4662 -
accuracy: 0.3763
Epoch 42/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4659 -
accuracy: 0.9604
Epoch 43/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4652 -
```

```
accuracy: 0.4517
Epoch 44/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4656 -
accuracy: 0.5371
Epoch 45/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4642 -
accuracy: 0.8169
Epoch 46/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4636 -
accuracy: 0.9603
Epoch 47/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4636 -
accuracy: 0.7731
Epoch 48/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4629 -
accuracy: 0.3519
Epoch 49/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4623 -
accuracy: 0.4261
Epoch 50/50
219/219 [==============================] - 1s 3ms/step - loss: 0.4617 -
accuracy: 0.0206
94/94 [==============================] - 1s 2ms/step - loss: 0.4584 - accuracy:
3.3333e-04

Test accuracy: 0.00033333332976326346

Test loss: 0.45844244956970215
```

Test accuracy: 0.968666672706604 after 50 epochs and test loss: 0.450

[69]:
```python
history = model_tfp.fit(np.asarray(X_train), np.asarray(y_train),epochs=50,␣
↪validation_split=0.3,  shuffle=True)
```

```
Epoch 1/50
154/154 [==============================] - 2s 8ms/step - loss: 0.4614 -
accuracy: 0.0253 - val_loss: 0.4622 - val_accuracy: 4.7619e-04
Epoch 2/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4616 -
accuracy: 0.3765 - val_loss: 0.4616 - val_accuracy: 0.9633
Epoch 3/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4608 -
accuracy: 0.3951 - val_loss: 0.4634 - val_accuracy: 0.9629
Epoch 4/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4599 -
accuracy: 0.9473 - val_loss: 0.4609 - val_accuracy: 0.9633
Epoch 5/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4602 -
```

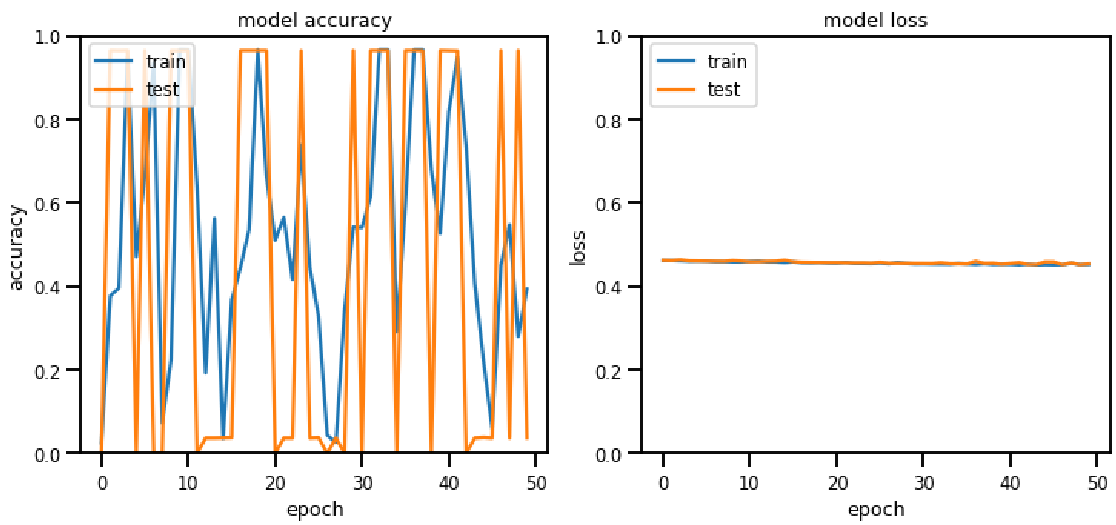
```
accuracy: 0.4706 - val_loss: 0.4606 - val_accuracy: 0.0000e+00
Epoch 6/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4599 -
accuracy: 0.6651 - val_loss: 0.4602 - val_accuracy: 0.9633
Epoch 7/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4586 -
accuracy: 0.9329 - val_loss: 0.4599 - val_accuracy: 0.0000e+00
Epoch 8/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4584 -
accuracy: 0.0729 - val_loss: 0.4597 - val_accuracy: 0.0000e+00
Epoch 9/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4580 -
accuracy: 0.2245 - val_loss: 0.4616 - val_accuracy: 0.9629
Epoch 10/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4576 -
accuracy: 0.9655 - val_loss: 0.4605 - val_accuracy: 0.9629
Epoch 11/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4592 -
accuracy: 0.9651 - val_loss: 0.4586 - val_accuracy: 0.9633
Epoch 12/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4599 -
accuracy: 0.6378 - val_loss: 0.4587 - val_accuracy: 0.0000e+00
Epoch 13/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4577 -
accuracy: 0.1920 - val_loss: 0.4595 - val_accuracy: 0.0367
Epoch 14/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4576 -
accuracy: 0.5624 - val_loss: 0.4601 - val_accuracy: 0.0367
Epoch 15/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4564 -
accuracy: 0.0359 - val_loss: 0.4624 - val_accuracy: 0.0371
Epoch 16/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4579 -
accuracy: 0.3671 - val_loss: 0.4586 - val_accuracy: 0.0376
Epoch 17/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4558 -
accuracy: 0.4439 - val_loss: 0.4572 - val_accuracy: 0.9633
Epoch 18/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4556 -
accuracy: 0.5347 - val_loss: 0.4569 - val_accuracy: 0.9633
Epoch 19/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4563 -
accuracy: 0.9655 - val_loss: 0.4567 - val_accuracy: 0.9633
Epoch 20/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4551 -
accuracy: 0.6629 - val_loss: 0.4564 - val_accuracy: 0.9633
Epoch 21/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4549 -
```

```
accuracy: 0.5096 - val_loss: 0.4564 - val_accuracy: 4.7619e-04
Epoch 22/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4565 -
accuracy: 0.5647 - val_loss: 0.4560 - val_accuracy: 0.0367
Epoch 23/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4543 -
accuracy: 0.4161 - val_loss: 0.4557 - val_accuracy: 0.0367
Epoch 24/50
154/154 [==============================] - 1s 4ms/step - loss: 0.4541 -
accuracy: 0.7382 - val_loss: 0.4558 - val_accuracy: 0.9629
Epoch 25/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4538 -
accuracy: 0.4449 - val_loss: 0.4552 - val_accuracy: 0.0367
Epoch 26/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4553 -
accuracy: 0.3273 - val_loss: 0.4572 - val_accuracy: 0.0376
Epoch 27/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4534 -
accuracy: 0.0431 - val_loss: 0.4548 - val_accuracy: 0.0000e+00
Epoch 28/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4566 -
accuracy: 0.0265 - val_loss: 0.4547 - val_accuracy: 0.0367
Epoch 29/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4542 -
accuracy: 0.3427 - val_loss: 0.4546 - val_accuracy: 0.0000e+00
Epoch 30/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4530 -
accuracy: 0.5414 - val_loss: 0.4544 - val_accuracy: 0.9633
Epoch 31/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4532 -
accuracy: 0.5398 - val_loss: 0.4543 - val_accuracy: 0.0000e+00
Epoch 32/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4528 -
accuracy: 0.6143 - val_loss: 0.4542 - val_accuracy: 0.9633
Epoch 33/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4526 -
accuracy: 0.9655 - val_loss: 0.4562 - val_accuracy: 0.9633
Epoch 34/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4524 -
accuracy: 0.9655 - val_loss: 0.4538 - val_accuracy: 0.9633
Epoch 35/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4541 -
accuracy: 0.2908 - val_loss: 0.4536 - val_accuracy: 0.0000e+00
Epoch 36/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4529 -
accuracy: 0.5937 - val_loss: 0.4534 - val_accuracy: 0.9633
Epoch 37/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4518 -
```

```
accuracy: 0.9655 - val_loss: 0.4598 - val_accuracy: 0.9633
Epoch 38/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4534 -
accuracy: 0.9655 - val_loss: 0.4546 - val_accuracy: 0.9629
Epoch 39/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4517 -
accuracy: 0.6786 - val_loss: 0.4550 - val_accuracy: 4.7619e-04
Epoch 40/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4521 -
accuracy: 0.5255 - val_loss: 0.4525 - val_accuracy: 0.9633
Epoch 41/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4518 -
accuracy: 0.8159 - val_loss: 0.4544 - val_accuracy: 0.9629
Epoch 42/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4506 -
accuracy: 0.9476 - val_loss: 0.4565 - val_accuracy: 0.9624
Epoch 43/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4523 -
accuracy: 0.7351 - val_loss: 0.4519 - val_accuracy: 0.0000e+00
Epoch 44/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4503 -
accuracy: 0.4059 - val_loss: 0.4519 - val_accuracy: 0.0367
Epoch 45/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4510 -
accuracy: 0.2210 - val_loss: 0.4583 - val_accuracy: 0.0381
Epoch 46/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4506 -
accuracy: 0.0594 - val_loss: 0.4582 - val_accuracy: 0.0367
Epoch 47/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4509 -
accuracy: 0.4496 - val_loss: 0.4513 - val_accuracy: 0.9633
Epoch 48/50
154/154 [==============================] - 1s 5ms/step - loss: 0.4551 -
accuracy: 0.5469 - val_loss: 0.4557 - val_accuracy: 0.0367
Epoch 49/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4513 -
accuracy: 0.2798 - val_loss: 0.4513 - val_accuracy: 0.9633
Epoch 50/50
154/154 [==============================] - 1s 6ms/step - loss: 0.4517 -
accuracy: 0.3935 - val_loss: 0.4534 - val_accuracy: 0.0371
```

[70]:
```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
```

```python
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.ylim(0, 1)
# summarize history for loss
plt.subplot(1,2,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.ylim(0, 1)
plt.show()
```



[71]:
```python
model_tfp.summary()
```

```
Model: "sequential_2"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_flipout (DenseFlipout  (None, 16)               368
 )

 dense_flipout_1 (DenseFlipo  (None, 6)                198
 ut)

 dense_flipout_2 (DenseFlipo  (None, 3)                39
 ut)

=================================================================
```

```
Total params: 605
Trainable params: 605
Non-trainable params: 0
_________________________________________________________________
```

define tensorboard variables for we plots

Train BNN with a small training subset.

Train BNN with the whole training set.

EXP VBNN:

[72]:
```python
dist = tfp.distributions
dataset_size = len(X_train)
kl_divergence_function = (lambda q, p, _: dist.kl_divergence(q, p) / tf.
 ↪cast(dataset_size, dtype=tf.float32))

model_tfp_v1 = tf.keras.Sequential([
    tf.keras.Input(X_train.shape[1]),
    tfp.layers.DenseFlipout(14, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu),
    tfp.layers.DenseFlipout(6, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu ),
    tfp.layers.DenseFlipout(2, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.softmax),
])

learning_rate = 0.002
model_tfp_v1.compile(optimizer=tf.keras.optimizers.
 ↪Adam(learning_rate),loss='binary_crossentropy',metrics=['accuracy'])
```

```
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:102: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:112: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
```

[74]:
```python
from keras.utils.vis_utils import plot_model
```

[75]:
```python
history = model_tfp_v1.fit(X_train, y_train, epochs=40)
test_loss, test_acc = model_tfp_v1.evaluate(X_test,  y_test)
```

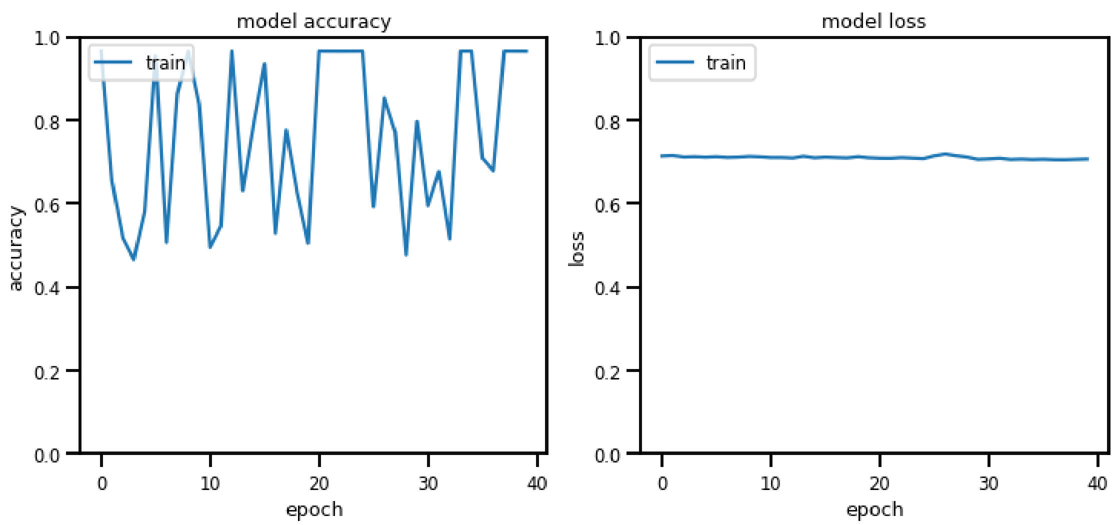
```python
print('\nTest accuracy:', test_acc)
print('\nTest loss:', test_loss)
# TRY REMOVING THE VALIDATION PART FROM THE FIT
# validation_data = (np.asarray(X_test), np.asarray(y_test))
#history = normal_bnn_model.fit(np.asarray(X_train), np.
 ↪asarray(y_train),epochs=50,validation_split=0.2, shuffle=True)
# to see history:
# list all data in history
print(history.history.keys())
# summarize history for accuracy

#normal_bnn_model.save('model_tfp_v1.h5')
#normal_bnn_model.save('saved_model/model_tfp_v1')
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'])
#plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.ylim(0, 1)
# summarize history for loss
plt.subplot(1,2,2)
plt.plot(history.history['loss'])
#plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.ylim(0, 1)
plt.show()
plot_model(model_tfp_v1, to_file='model_plot.png', show_shapes=True,␣
 ↪show_layer_names=True)
```

```
Epoch 1/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7136 -
accuracy: 0.9647
Epoch 2/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7148 -
accuracy: 0.6534
Epoch 3/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7111 -
accuracy: 0.5171
Epoch 4/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7119 -
accuracy: 0.4654
```

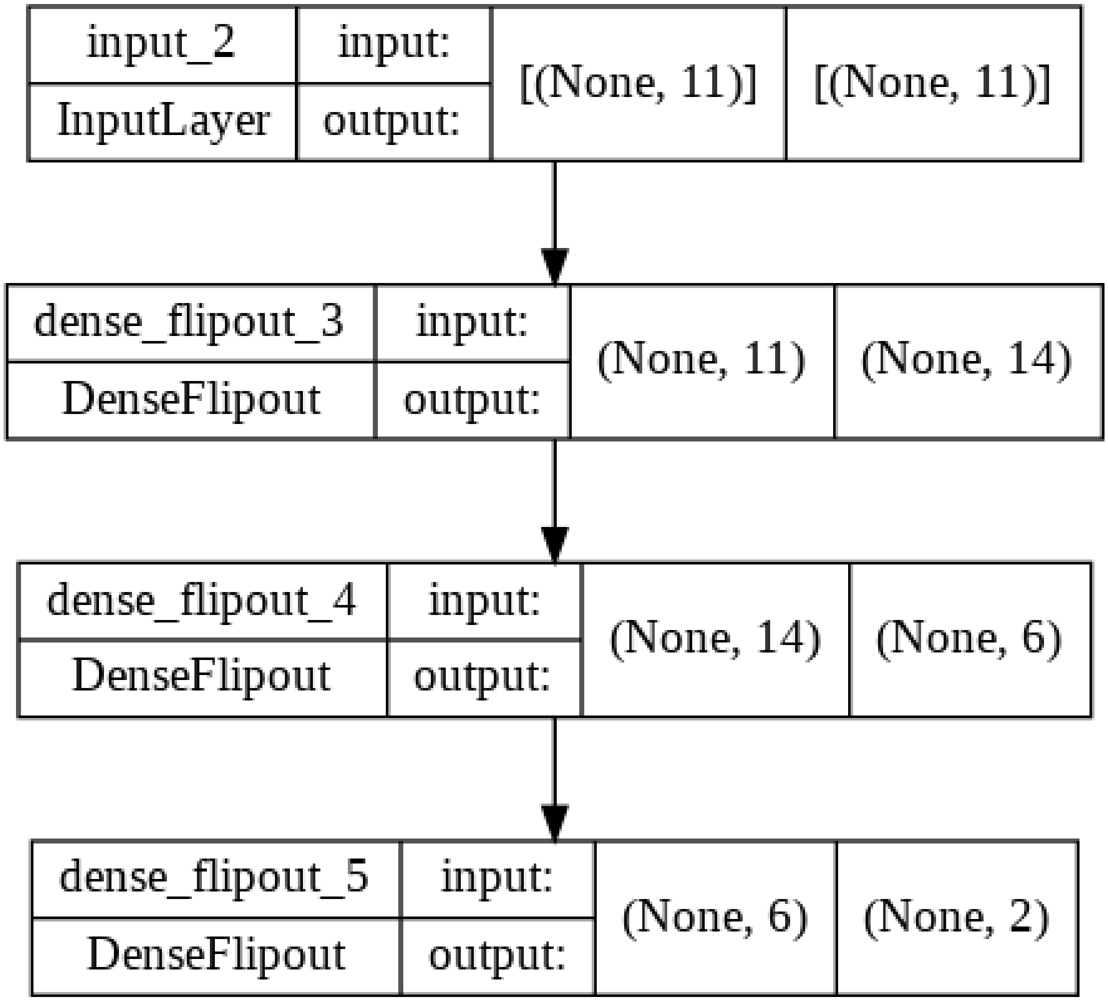
```
Epoch 5/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7107 -
accuracy: 0.5811
Epoch 6/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7119 -
accuracy: 0.9533
Epoch 7/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7102 -
accuracy: 0.5066
Epoch 8/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7109 -
accuracy: 0.8629
Epoch 9/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7125 -
accuracy: 0.9647
Epoch 10/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7116 -
accuracy: 0.8380
Epoch 11/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7099 -
accuracy: 0.4944
Epoch 12/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7099 -
accuracy: 0.5456
Epoch 13/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7087 -
accuracy: 0.9649
Epoch 14/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7129 -
accuracy: 0.6311
Epoch 15/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7092 -
accuracy: 0.7903
Epoch 16/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7109 -
accuracy: 0.9346
Epoch 17/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7099 -
accuracy: 0.5289
Epoch 18/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7090 -
accuracy: 0.7761
Epoch 19/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7120 -
accuracy: 0.6250
Epoch 20/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7094 -
accuracy: 0.5046
```

```
Epoch 21/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7082 -
accuracy: 0.9649
Epoch 22/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7079 -
accuracy: 0.9647
Epoch 23/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7097 -
accuracy: 0.9646
Epoch 24/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7084 -
accuracy: 0.9647
Epoch 25/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7072 -
accuracy: 0.9649
Epoch 26/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7139 -
accuracy: 0.5920
Epoch 27/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7185 -
accuracy: 0.8526
Epoch 28/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7143 -
accuracy: 0.7704
Epoch 29/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7110 -
accuracy: 0.4771
Epoch 30/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7054 -
accuracy: 0.7960
Epoch 31/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7065 -
accuracy: 0.5941
Epoch 32/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7080 -
accuracy: 0.6764
Epoch 33/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7051 -
accuracy: 0.5149
Epoch 34/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7060 -
accuracy: 0.9647
Epoch 35/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7048 -
accuracy: 0.9649
Epoch 36/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7056 -
accuracy: 0.7081
```

```
Epoch 37/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7046 -
accuracy: 0.6783
Epoch 38/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7044 -
accuracy: 0.9649
Epoch 39/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7053 -
accuracy: 0.9647
Epoch 40/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7061 -
accuracy: 0.9646
94/94 [==============================] - 0s 3ms/step - loss: 0.7133 - accuracy:
0.9683

Test accuracy: 0.9683333039283752

Test loss: 0.7132568955421448
dict_keys(['loss', 'accuracy'])
```



[75]:



[76]:
```
model_tfp_v1.summary()
```

```
Model: "sequential_3"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_flipout_3 (DenseFlipo  (None, 14)               322
 ut)

 dense_flipout_4 (DenseFlipo  (None, 6)                174
 ut)

 dense_flipout_5 (DenseFlipo  (None, 2)                26
 ut)

=================================================================
```

```
Total params: 522
Trainable params: 522
Non-trainable params: 0
_________________________________________________________________
```

[77]:
```
!pip install pickle5
```

```
Collecting pickle5
  Downloading
pickle5-0.0.12-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.whl (256 kB)
     |                 | 256 kB 4.4 MB/s
Installing collected packages: pickle5
Successfully installed pickle5-0.0.12
```

[78]:
```
import pickle
filename = 'model_tfp1v1.pkl'
tf.saved_model.SaveOptions(
    namespace_whitelist=None, save_debug_info=False, function_aliases=None,
    experimental_io_device=None, experimental_variable_policy=None,
    experimental_custom_gradients=True
)
pickle.dump(model_tfp_v1, open(filename, 'wb'))
```

```
INFO:tensorflow:Assets written to:
ram://b9d4cb47-3e23-4ed1-ace3-814d658df77f/assets
```

[79]:
```
!mkdir -p saved_model
```

[80]:
```
#saving tensorflow model of version v1 to drive. download this and place it in
#streamlit local folder and load it using tensorflow load model
model_tfp_v1.save('saved_model/model_tfp_v1')
```

```
INFO:tensorflow:Assets written to: saved_model/model_tfp_v1/assets
```

[81]:
```
#saving model into hdf5 format and load the same file using same loadmodel␣
 ↪function
model_tfp_v1.save('model_tfp_v1.h5')
```

[82]:
```
# use this to load the model into local
new_model = tf.keras.models.load_model('saved_model/model_tfp_v1')

# Check its architecture
new_model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

 dense_flipout_3 (DenseFlipo (None, 14)   322
 ut)

 dense_flipout_4 (DenseFlipo (None, 6)   174
 ut)

 dense_flipout_5 (DenseFlipo (None, 2)   26
 ut)

===
Total params: 522
Trainable params: 522
Non-trainable params: 0
____

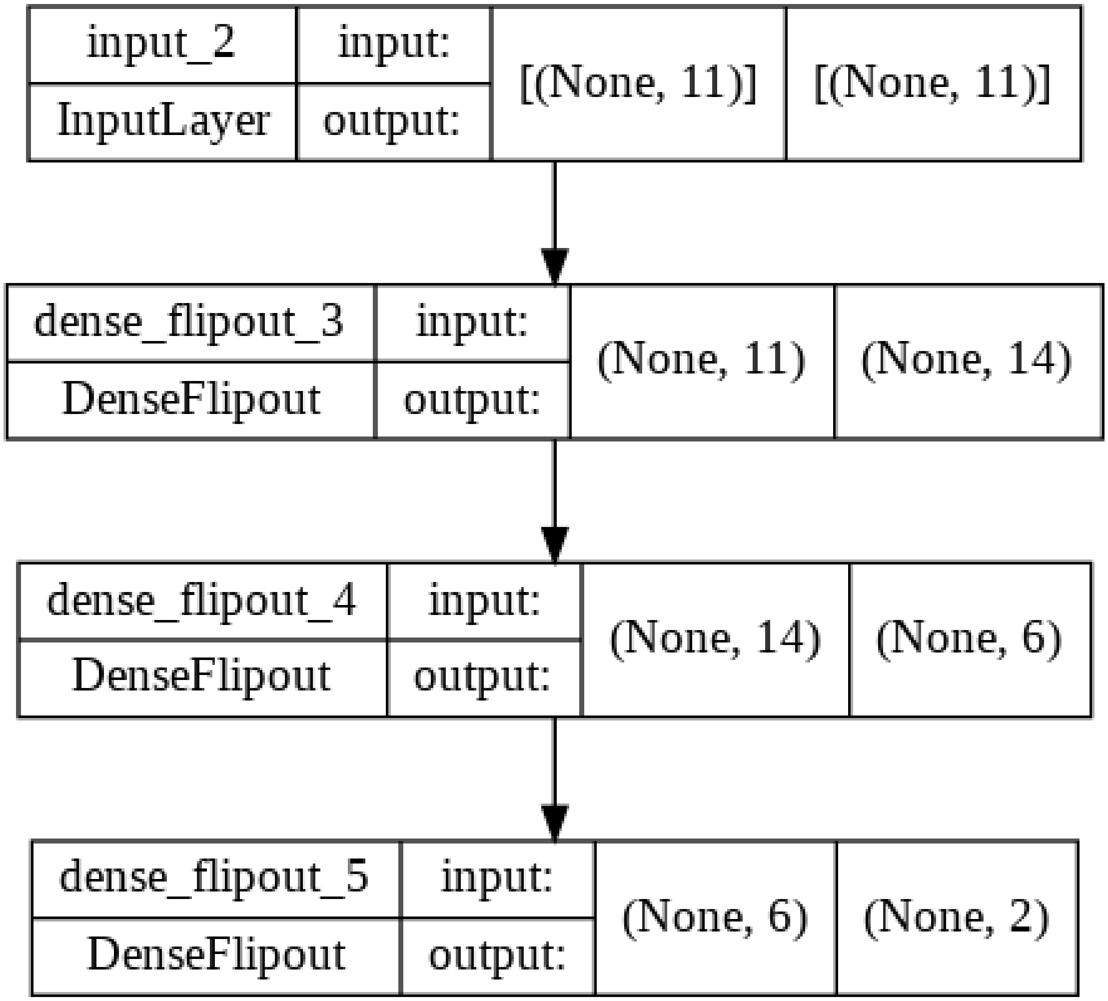
[83]:
```
!pip3 install ann_visualizer
!pip install graphviz
```

```
Collecting ann_visualizer
  Downloading ann_visualizer-2.5.tar.gz (4.7 kB)
Building wheels for collected packages: ann-visualizer
  Building wheel for ann-visualizer (setup.py) … done
  Created wheel for ann-visualizer: filename=ann_visualizer-2.5-py3-none-any.whl
size=4168
sha256=c2dcf73d0b5b3d199892db2765cf3d79293cf3b88bd5a674925769aa286c8d8a
  Stored in directory: /root/.cache/pip/wheels/1b/fc/58/2ab1c3b30350105929308bec
ddda4fb59b1358e54f985e1f4a
Successfully built ann-visualizer
Installing collected packages: ann-visualizer
Successfully installed ann-visualizer-2.5
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-
packages (0.10.1)
```

[84]:
```
from ann_visualizer.visualize import ann_viz;

#ann_viz(new_model, title="My first neural network")
```

[85]:
```
from keras.utils.vis_utils import plot_model
plot_model(new_model, to_file='model_plot1.png', show_shapes=True,␣
 ↪show_layer_names=True)
```

[85]:



v2

[87]:
```
dist = tfp.distributions
dataset_size = len(X_train)
kl_divergence_function = (lambda q, p, _: dist.kl_divergence(q, p) / tf.cast(dataset_size, dtype=tf.float32))

model_tfp_v2 = tf.keras.Sequential([
    tf.keras.Input(X_train.shape[1]),
    tfp.layers.DenseFlipout(14, kernel_divergence_fn=kl_divergence_function, activation=tf.nn.relu),
    tfp.layers.DenseFlipout(6, kernel_divergence_fn=kl_divergence_function, activation=tf.nn.relu ),
    tfp.layers.DenseFlipout(2, kernel_divergence_fn=kl_divergence_function, activation=tf.nn.softmax),
```

```
])

learning_rate = 0.005
model_tfp_v2.compile(optimizer=tf.keras.optimizers.
 ↪Adam(learning_rate),loss='binary_crossentropy',metrics=['accuracy'])

history = model_tfp_v2.fit(X_train, y_train, epochs=80)
test_loss, test_acc = model_tfp_v2.evaluate(X_test,  y_test)
print('\nTest accuracy:', test_acc)
print('\nTest loss:', test_loss)

# TRY REMOVING THE VALIDATION PART FROM THE FIT
# validation_data = (np.asarray(X_test), np.asarray(y_test))
#history = normal_bnn_model.fit(np.asarray(X_train), np.
 ↪asarray(y_train),epochs=50,validation_split=0.2, shuffle=True)
# to see history:
# list all data in history
print(history.history.keys())
# summarize history for accuracy

model_tfp_v2.save('model_tfp_v2.h5')
model_tfp_v2.save('saved_model/model_tfp_v2')
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'])
#plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.ylim(0, 1)
# summarize history for loss
plt.subplot(1,2,2)
plt.plot(history.history['loss'])
#plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.ylim(0, 1)
plt.show()
plot_model(model_tfp_v2, to_file='model_plot.png', show_shapes=True,␣
 ↪show_layer_names=True)

model_tfp_v2.summary()
```

```
/usr/local/lib/python3.7/dist-
```

```
packages/tensorflow_probability/python/layers/util.py:102: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:112: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
```

```
Epoch 1/80
219/219 [==============================] - 3s 3ms/step - loss: 0.7955 -
accuracy: 0.5663
Epoch 2/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7583 -
accuracy: 0.5519
Epoch 3/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7471 -
accuracy: 0.4826
Epoch 4/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7377 -
accuracy: 0.5129
Epoch 5/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7334 -
accuracy: 0.5441
Epoch 6/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7321 -
accuracy: 0.5050
Epoch 7/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7292 -
accuracy: 0.4769
Epoch 8/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7271 -
accuracy: 0.4987
Epoch 9/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7244 -
accuracy: 0.5139
Epoch 10/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7256 -
accuracy: 0.5439
Epoch 11/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7249 -
accuracy: 0.4936
Epoch 12/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7203 -
accuracy: 0.5624
Epoch 13/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7200 -
```

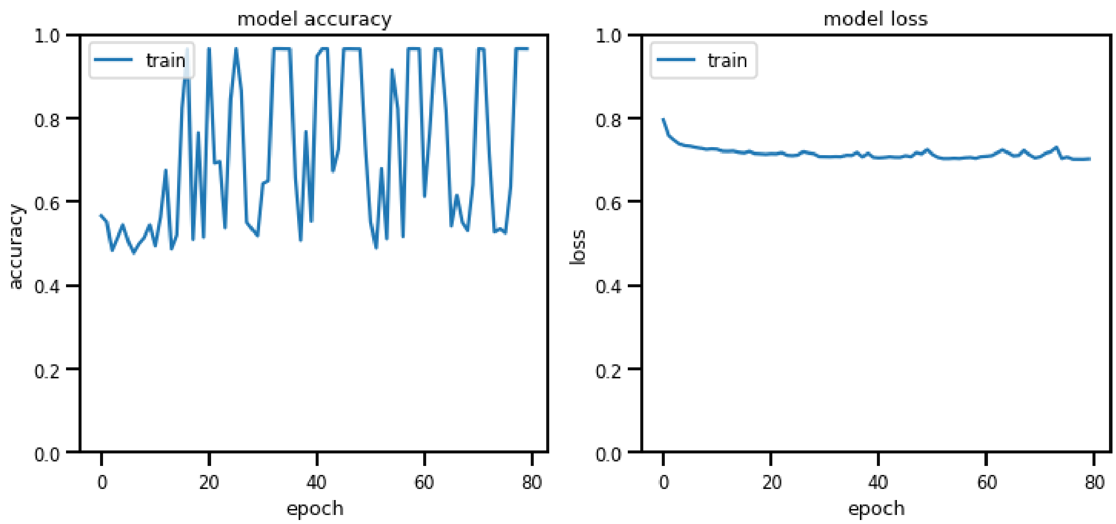
```
accuracy: 0.6741
Epoch 14/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7207 -
accuracy: 0.4863
Epoch 15/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7178 -
accuracy: 0.5201
Epoch 16/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7161 -
accuracy: 0.8223
Epoch 17/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7198 -
accuracy: 0.9640
Epoch 18/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7144 -
accuracy: 0.5094
Epoch 19/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7138 -
accuracy: 0.7636
Epoch 20/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7128 -
accuracy: 0.5149
Epoch 21/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7139 -
accuracy: 0.9647
Epoch 22/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7136 -
accuracy: 0.6917
Epoch 23/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7164 -
accuracy: 0.6956
Epoch 24/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7105 -
accuracy: 0.5369
Epoch 25/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7096 -
accuracy: 0.8437
Epoch 26/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7109 -
accuracy: 0.9649
Epoch 27/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7190 -
accuracy: 0.8626
Epoch 28/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7158 -
accuracy: 0.5501
Epoch 29/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7137 -
```

```
accuracy: 0.5334
Epoch 30/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7072 -
accuracy: 0.5183
Epoch 31/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7070 -
accuracy: 0.6427
Epoch 32/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7067 -
accuracy: 0.6494
Epoch 33/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7073 -
accuracy: 0.9649
Epoch 34/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7070 -
accuracy: 0.9649
Epoch 35/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7106 -
accuracy: 0.9646
Epoch 36/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7102 -
accuracy: 0.9644
Epoch 37/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7170 -
accuracy: 0.6571
Epoch 38/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7066 -
accuracy: 0.5074
Epoch 39/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7152 -
accuracy: 0.7663
Epoch 40/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7056 -
accuracy: 0.5529
Epoch 41/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7044 -
accuracy: 0.9474
Epoch 42/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7052 -
accuracy: 0.9647
Epoch 43/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7069 -
accuracy: 0.9647
Epoch 44/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7057 -
accuracy: 0.6729
Epoch 45/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7056 -
```

```
accuracy: 0.7246
Epoch 46/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7093 -
accuracy: 0.9643
Epoch 47/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7070 -
accuracy: 0.9646
Epoch 48/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7167 -
accuracy: 0.9643
Epoch 49/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7134 -
accuracy: 0.9641
Epoch 50/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7239 -
accuracy: 0.7347
Epoch 51/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7122 -
accuracy: 0.5510
Epoch 52/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7055 -
accuracy: 0.4887
Epoch 53/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7026 -
accuracy: 0.6784
Epoch 54/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7026 -
accuracy: 0.5111
Epoch 55/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7035 -
accuracy: 0.9143
Epoch 56/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7028 -
accuracy: 0.8207
Epoch 57/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7045 -
accuracy: 0.5164
Epoch 58/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7054 -
accuracy: 0.9647
Epoch 59/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7032 -
accuracy: 0.9649
Epoch 60/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7070 -
accuracy: 0.9644
Epoch 61/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7080 -
```

```
accuracy: 0.6129
Epoch 62/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7097 -
accuracy: 0.7707
Epoch 63/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7164 -
accuracy: 0.9641
Epoch 64/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7231 -
accuracy: 0.9636
Epoch 65/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7164 -
accuracy: 0.8139
Epoch 66/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7091 -
accuracy: 0.5413
Epoch 67/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7102 -
accuracy: 0.6149
Epoch 68/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7222 -
accuracy: 0.5503
Epoch 69/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7112 -
accuracy: 0.5301
Epoch 70/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7042 -
accuracy: 0.6401
Epoch 71/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7070 -
accuracy: 0.9647
Epoch 72/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7148 -
accuracy: 0.9636
Epoch 73/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7197 -
accuracy: 0.7181
Epoch 74/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7296 -
accuracy: 0.5270
Epoch 75/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7029 -
accuracy: 0.5346
Epoch 76/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7063 -
accuracy: 0.5247
Epoch 77/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7011 -
```

```
accuracy: 0.6351
Epoch 78/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7010 -
accuracy: 0.9649
Epoch 79/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7009 -
accuracy: 0.9649
Epoch 80/80
219/219 [==============================] - 1s 3ms/step - loss: 0.7018 -
accuracy: 0.9647
94/94 [==============================] - 1s 2ms/step - loss: 0.7030 - accuracy:
0.9690

Test accuracy: 0.968999981880188

Test loss: 0.7030236721038818
dict_keys(['loss', 'accuracy'])
INFO:tensorflow:Assets written to: saved_model/model_tfp_v2/assets
```



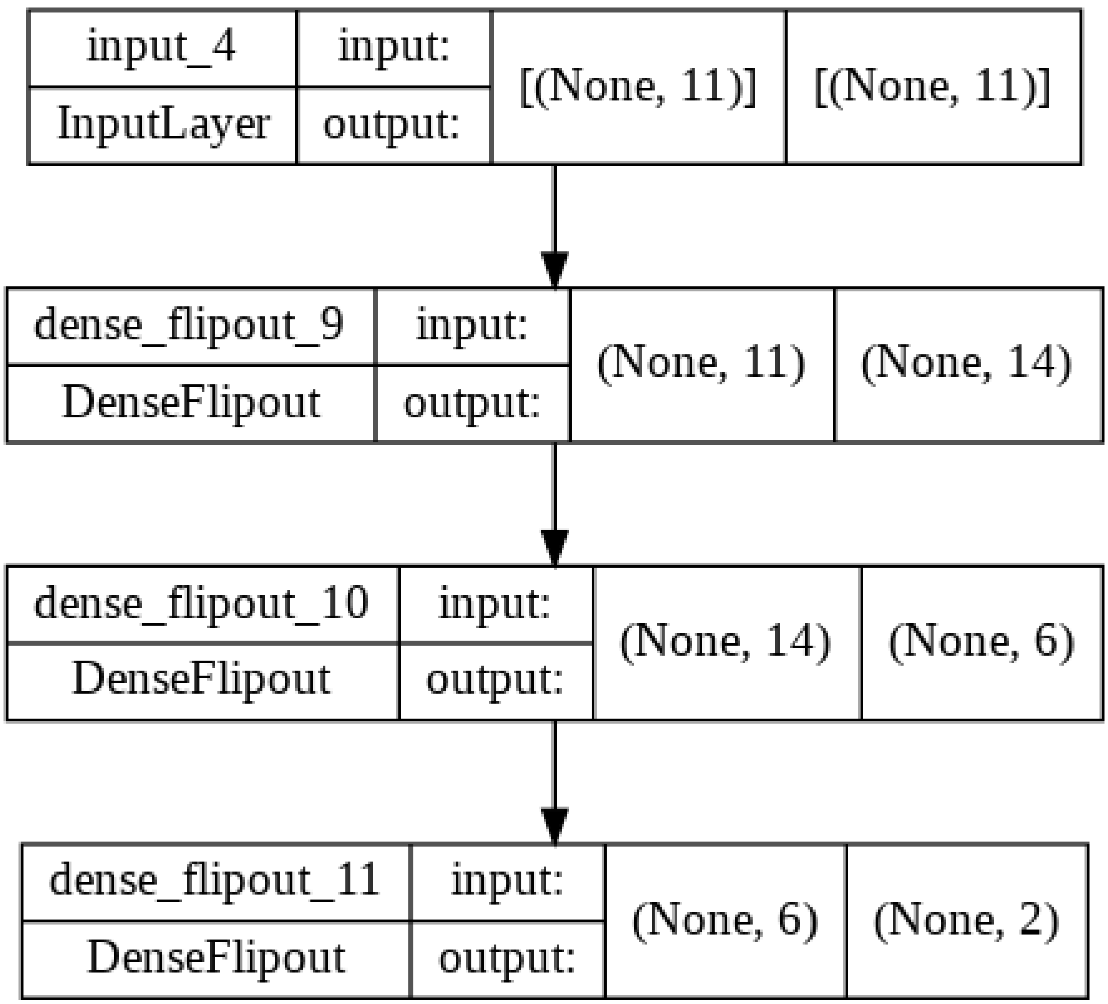

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_flipout_9 (DenseFlipo  (None, 14)               322
 ut)

 dense_flipout_10 (DenseFlip  (None, 6)                174
 out)

 dense_flipout_11 (DenseFlip  (None, 2)                26
```

```
out)

=================================================================
Total params: 522
Trainable params: 522
Non-trainable params: 0
_________________________________________________________________
```

[88]:
```
#ann_viz(model_tfp_v2, title="My Second neural network")
```

[89]:
```
from keras.utils.vis_utils import plot_model
plot_model(model_tfp_v2, to_file='model_plot.png', show_shapes=True,␣
 ↪show_layer_names=True)
```

[89]:



visualize BNN

[90]:
```
!pip3 install keras
!pip3 install ann_visualizer
!pip install graphviz
```

```
Requirement already satisfied: keras in /usr/local/lib/python3.7/dist-packages
(2.8.0)
Requirement already satisfied: ann_visualizer in /usr/local/lib/python3.7/dist-
packages (2.5)
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-
packages (0.10.1)
```

```
Experiment 3: probabilistic Bayesian neural network: not needed
```

0.0.3 DIFFERENT BNN'S

```
1. NORMAL BNN
2. BNN WITH DIFFERENT DROPOUTS
3. BNN WITH DIFFERENT EARLY STOPS
4. BNN WITH DIFFERENT REGULARIZERS
5. SIR mentioned to work on transformers also
6. MIXING OF THE ABOVE VARIANTS AND COMPARING WITH THE NORMAL ANN
```

```
PLOT THE UNCERTAINITIES FOR ALL THESE MODELS
```

1. NORMAL BNN

[91]:
```
dist = tfp.distributions
dataset_size = len(X_train)
kl_divergence_function = (lambda q, p, _: dist.kl_divergence(q, p) / tf.
 ↪cast(dataset_size, dtype=tf.float32))

normal_bnn_model = tf.keras.Sequential([
    tf.keras.Input(X_train.shape[1]),
    tfp.layers.DenseFlipout(16, kernel_divergence_fn=kl_divergence_function␣
 ↪),#activation=tf.nn.relu),
    tfp.layers.DenseFlipout(6,␣
 ↪kernel_divergence_fn=kl_divergence_function,activation=tf.nn.relu),
    tfp.layers.DenseFlipout(3, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.softmax),
])

learning_rate = 0.001
normal_bnn_model.compile(optimizer=tf.keras.optimizers.
 ↪Adam(learning_rate),loss='binary_crossentropy',metrics=['accuracy'])
```

```
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:102: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
```

```
Please use `layer.add_weight` method instead.
  trainable=trainable)
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:112: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
```

[92]:

```
# TRY REMOVING THE VALIDATION PART FROM THE FIT
# validation_data = (np.asarray(X_test), np.asarray(y_test))
history = normal_bnn_model.fit(np.asarray(X_train), np.
 ↪asarray(y_train),epochs=50,validation_split=0.2, shuffle=True)
# to see history:
# list all data in history
print(history.history.keys())
# summarize history for accuracy
test_loss, test_acc = normal_bnn_model.evaluate(X_test,  y_test)
print('\nTest accuracy:', test_acc)
print('\nTest loss:', test_loss)

normal_bnn_model.save('normal_bnn_model.h5')
normal_bnn_model.save('saved_model/normal_bnn_model')
```

```
Epoch 1/50
175/175 [==============================] - 3s 7ms/step - loss: 0.8485 -
accuracy: 0.0720 - val_loss: 0.5491 - val_accuracy: 0.0136
Epoch 2/50
175/175 [==============================] - 1s 4ms/step - loss: 0.5352 -
accuracy: 0.0082 - val_loss: 0.5369 - val_accuracy: 0.0050
Epoch 3/50
175/175 [==============================] - 1s 4ms/step - loss: 0.5295 -
accuracy: 0.0045 - val_loss: 0.5322 - val_accuracy: 0.0021
Epoch 4/50
175/175 [==============================] - 1s 4ms/step - loss: 0.5265 -
accuracy: 0.0021 - val_loss: 0.5283 - val_accuracy: 0.0014
Epoch 5/50
175/175 [==============================] - 1s 4ms/step - loss: 0.5229 -
accuracy: 0.0020 - val_loss: 0.5257 - val_accuracy: 0.0021
Epoch 6/50
175/175 [==============================] - 1s 4ms/step - loss: 0.5203 -
accuracy: 0.0493 - val_loss: 0.5233 - val_accuracy: 0.0429
Epoch 7/50
175/175 [==============================] - 1s 3ms/step - loss: 0.5181 -
accuracy: 0.9146 - val_loss: 0.5205 - val_accuracy: 0.9564
Epoch 8/50
175/175 [==============================] - 1s 4ms/step - loss: 0.5149 -
accuracy: 0.5004 - val_loss: 0.5183 - val_accuracy: 0.0400
Epoch 9/50
```

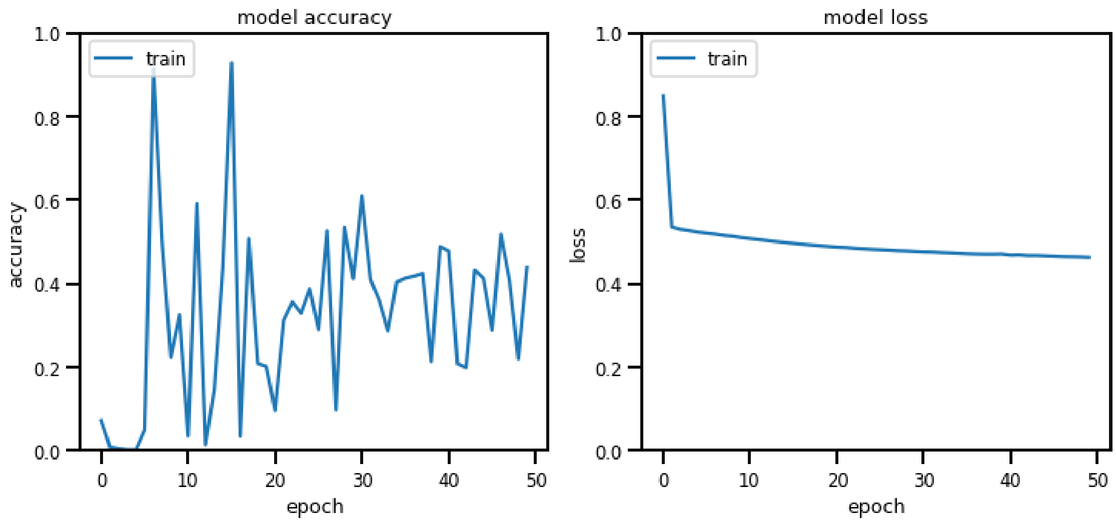
```
175/175 [==============================] - 1s 4ms/step - loss: 0.5130 -
accuracy: 0.2229 - val_loss: 0.5167 - val_accuracy: 0.9571
Epoch 10/50
175/175 [==============================] - 1s 3ms/step - loss: 0.5098 -
accuracy: 0.3250 - val_loss: 0.5126 - val_accuracy: 0.0400
Epoch 11/50
175/175 [==============================] - 1s 4ms/step - loss: 0.5072 -
accuracy: 0.0350 - val_loss: 0.5101 - val_accuracy: 0.0407
Epoch 12/50
175/175 [==============================] - 1s 3ms/step - loss: 0.5049 -
accuracy: 0.5905 - val_loss: 0.5077 - val_accuracy: 0.0407
Epoch 13/50
175/175 [==============================] - 1s 3ms/step - loss: 0.5024 -
accuracy: 0.0139 - val_loss: 0.5053 - val_accuracy: 0.0000e+00
Epoch 14/50
175/175 [==============================] - 1s 3ms/step - loss: 0.5000 -
accuracy: 0.1459 - val_loss: 0.5030 - val_accuracy: 0.0400
Epoch 15/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4977 -
accuracy: 0.4389 - val_loss: 0.5008 - val_accuracy: 0.9593
Epoch 16/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4957 -
accuracy: 0.9271 - val_loss: 0.4988 - val_accuracy: 0.0407
Epoch 17/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4936 -
accuracy: 0.0343 - val_loss: 0.4969 - val_accuracy: 0.0407
Epoch 18/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4917 -
accuracy: 0.5071 - val_loss: 0.4952 - val_accuracy: 0.0000e+00
Epoch 19/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4901 -
accuracy: 0.2086 - val_loss: 0.4934 - val_accuracy: 7.1429e-04
Epoch 20/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4883 -
accuracy: 0.2014 - val_loss: 0.4917 - val_accuracy: 0.9600
Epoch 21/50
175/175 [==============================] - 1s 4ms/step - loss: 0.4867 -
accuracy: 0.0957 - val_loss: 0.4901 - val_accuracy: 0.0000e+00
Epoch 22/50
175/175 [==============================] - 1s 4ms/step - loss: 0.4857 -
accuracy: 0.3114 - val_loss: 0.4888 - val_accuracy: 0.9600
Epoch 23/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4839 -
accuracy: 0.3555 - val_loss: 0.4874 - val_accuracy: 0.0021
Epoch 24/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4826 -
accuracy: 0.3282 - val_loss: 0.4864 - val_accuracy: 0.9579
Epoch 25/50
```

```
175/175 [==============================] - 1s 4ms/step - loss: 0.4814 -
accuracy: 0.3870 - val_loss: 0.4851 - val_accuracy: 0.0407
Epoch 26/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4805 -
accuracy: 0.2895 - val_loss: 0.4838 - val_accuracy: 0.9593
Epoch 27/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4794 -
accuracy: 0.5261 - val_loss: 0.4828 - val_accuracy: 7.1429e-04
Epoch 28/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4781 -
accuracy: 0.0973 - val_loss: 0.4818 - val_accuracy: 0.9600
Epoch 29/50
175/175 [==============================] - 1s 4ms/step - loss: 0.4775 -
accuracy: 0.5346 - val_loss: 0.4810 - val_accuracy: 0.9600
Epoch 30/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4763 -
accuracy: 0.4112 - val_loss: 0.4800 - val_accuracy: 0.0400
Epoch 31/50
175/175 [==============================] - 1s 4ms/step - loss: 0.4753 -
accuracy: 0.6093 - val_loss: 0.4789 - val_accuracy: 0.0400
Epoch 32/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4748 -
accuracy: 0.4091 - val_loss: 0.4778 - val_accuracy: 0.0400
Epoch 33/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4737 -
accuracy: 0.3587 - val_loss: 0.4775 - val_accuracy: 7.1429e-04
Epoch 34/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4729 -
accuracy: 0.2861 - val_loss: 0.4799 - val_accuracy: 7.1429e-04
Epoch 35/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4719 -
accuracy: 0.4032 - val_loss: 0.4755 - val_accuracy: 0.9607
Epoch 36/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4707 -
accuracy: 0.4121 - val_loss: 0.4745 - val_accuracy: 0.0000e+00
Epoch 37/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4701 -
accuracy: 0.4170 - val_loss: 0.4739 - val_accuracy: 0.0400
Epoch 38/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4697 -
accuracy: 0.4236 - val_loss: 0.4738 - val_accuracy: 0.0400
Epoch 39/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4696 -
accuracy: 0.2130 - val_loss: 0.4725 - val_accuracy: 0.0400
Epoch 40/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4700 -
accuracy: 0.4879 - val_loss: 0.4719 - val_accuracy: 0.0000e+00
Epoch 41/50
```

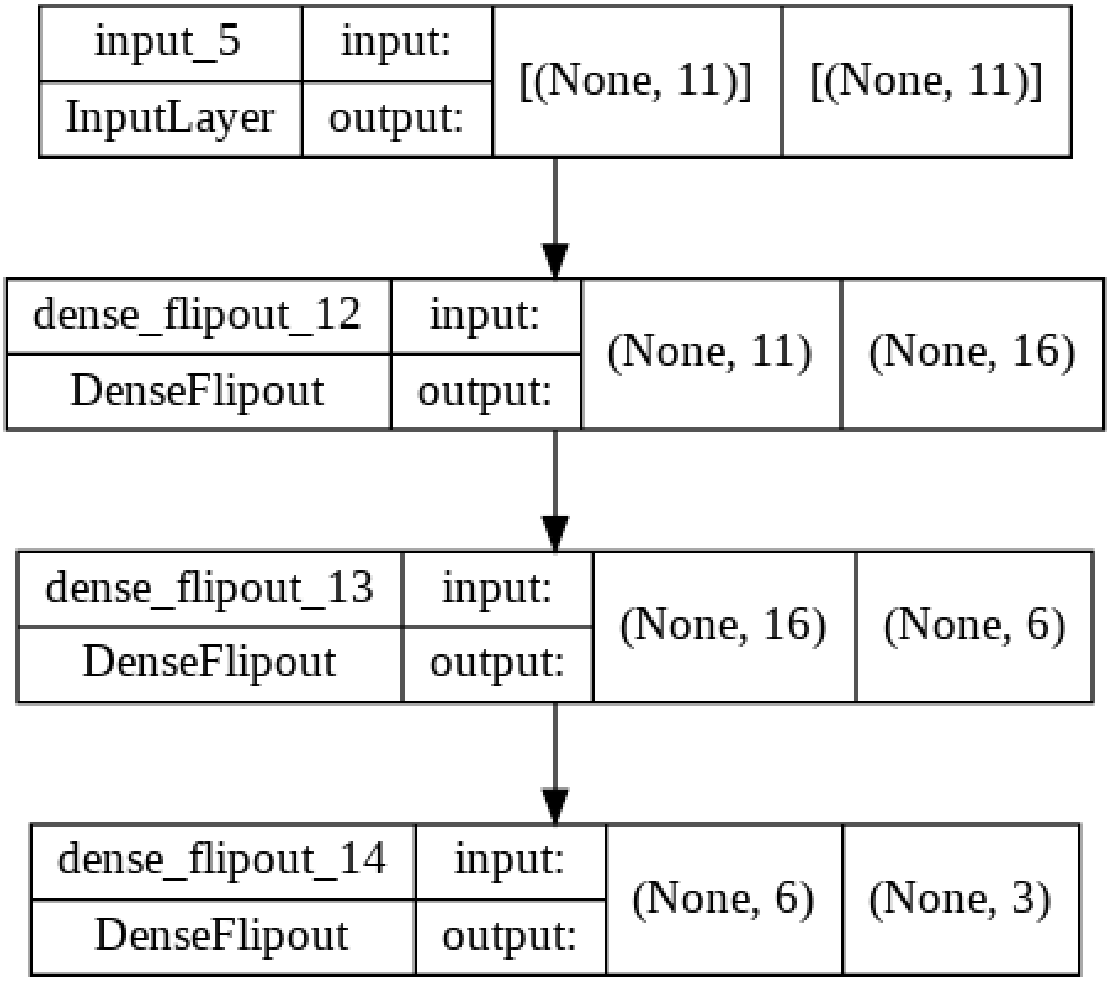
```
175/175 [==============================] - 1s 3ms/step - loss: 0.4674 -
accuracy: 0.4775 - val_loss: 0.4712 - val_accuracy: 0.0400
Epoch 42/50
175/175 [==============================] - 1s 4ms/step - loss: 0.4680 -
accuracy: 0.2079 - val_loss: 0.4706 - val_accuracy: 0.0000e+00
Epoch 43/50
175/175 [==============================] - 1s 4ms/step - loss: 0.4663 -
accuracy: 0.1984 - val_loss: 0.4700 - val_accuracy: 0.0400
Epoch 44/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4664 -
accuracy: 0.4320 - val_loss: 0.4693 - val_accuracy: 0.9600
Epoch 45/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4655 -
accuracy: 0.4114 - val_loss: 0.4687 - val_accuracy: 0.0400
Epoch 46/50
175/175 [==============================] - 1s 4ms/step - loss: 0.4647 -
accuracy: 0.2880 - val_loss: 0.4686 - val_accuracy: 0.9607
Epoch 47/50
175/175 [==============================] - 1s 4ms/step - loss: 0.4637 -
accuracy: 0.5179 - val_loss: 0.4676 - val_accuracy: 0.0400
Epoch 48/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4634 -
accuracy: 0.4064 - val_loss: 0.4670 - val_accuracy: 0.0000e+00
Epoch 49/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4630 -
accuracy: 0.2191 - val_loss: 0.4665 - val_accuracy: 0.9600
Epoch 50/50
175/175 [==============================] - 1s 3ms/step - loss: 0.4621 -
accuracy: 0.4387 - val_loss: 0.4660 - val_accuracy: 0.9600
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
94/94 [==============================] - 1s 3ms/step - loss: 0.4607 - accuracy:
0.9690

Test accuracy: 0.968999981880188

Test loss: 0.4606759548187256
INFO:tensorflow:Assets written to: saved_model/normal_bnn_model/assets
```

[93]:
```python
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'])
#plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.ylim(0, 1)
```

```
# summarize history for loss
plt.subplot(1,2,2)
plt.plot(history.history['loss'])
#plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.ylim(0, 1)
plt.show()
plot_model(normal_bnn_model, to_file='model_plot.png', show_shapes=True,␣
 ↪show_layer_names=True)
```



[93]:



NORMAL BNN2

[94]:
```
dist = tfp.distributions
dataset_size = len(X_train)
kl_divergence_function = (lambda q, p, _: dist.kl_divergence(q, p) / tf.
 ↪cast(dataset_size, dtype=tf.float32))

normal_bnn2_model = tf.keras.Sequential([
    tf.keras.Input(X_train.shape[1]),
    #Dense(units = 22, activation = 'relu'),
    tfp.layers.DenseFlipout(22,␣
 ↪kernel_divergence_fn=kl_divergence_function),#activation=tf.nn.relu),
    tfp.layers.DenseFlipout(12,␣
 ↪kernel_divergence_fn=kl_divergence_function),#activation=tf.nn.relu),
    tfp.layers.DenseFlipout(4, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu),
```

```
    tfp.layers.DenseFlipout(2, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.softmax),
])

learning_rate = 0.00065
normal_bnn2_model.compile(optimizer=tf.keras.optimizers.
 ↪Adam(learning_rate),loss='binary_crossentropy',metrics=['accuracy'])
```

```
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:102: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:112: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
```

[95]:
```
# TRY REMOVING THE VALIDATION PART FROM THE FIT
# validation_data = (np.asarray(X_test), np.asarray(y_test))
history = normal_bnn2_model.fit(np.asarray(X_train), np.
 ↪asarray(y_train),epochs=100)#,validation_split=0.2, shuffle=True)
# to see history:
# list all data in history
print(history.history.keys())
# summarize history for accuracy
test_loss, test_acc = normal_bnn_model.evaluate(X_test,  y_test)
print('\nTest accuracy:', test_acc)
print('\nTest loss:', test_loss)
```

```
Epoch 1/100
219/219 [==============================] - 4s 4ms/step - loss: 1.0113 -
accuracy: 0.1489
Epoch 2/100
219/219 [==============================] - 1s 4ms/step - loss: 0.8941 -
accuracy: 0.0466
Epoch 3/100
219/219 [==============================] - 1s 4ms/step - loss: 0.8894 -
accuracy: 0.0410
Epoch 4/100
219/219 [==============================] - 1s 4ms/step - loss: 0.8851 -
accuracy: 0.0590
Epoch 5/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8821 -
accuracy: 0.0371
Epoch 6/100
```

```
219/219 [==============================] - 1s 3ms/step - loss: 0.8772 -
accuracy: 0.4076
Epoch 7/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8730 -
accuracy: 0.5207
Epoch 8/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8688 -
accuracy: 0.3263
Epoch 9/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8644 -
accuracy: 0.5873
Epoch 10/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8599 -
accuracy: 0.6797
Epoch 11/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8558 -
accuracy: 0.4079
Epoch 12/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8504 -
accuracy: 0.3676
Epoch 13/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8457 -
accuracy: 0.6019
Epoch 14/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8407 -
accuracy: 0.5616
Epoch 15/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8357 -
accuracy: 0.4156
Epoch 16/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8305 -
accuracy: 0.3644
Epoch 17/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8258 -
accuracy: 0.5493
Epoch 18/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8208 -
accuracy: 0.4597
Epoch 19/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8161 -
accuracy: 0.4830
Epoch 20/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8113 -
accuracy: 0.4094
Epoch 21/100
219/219 [==============================] - 1s 3ms/step - loss: 0.8067 -
accuracy: 0.4781
Epoch 22/100
```

```
219/219 [==============================] - 1s 3ms/step - loss: 0.8030 -
accuracy: 0.4650
Epoch 23/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7988 -
accuracy: 0.5200
Epoch 24/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7954 -
accuracy: 0.4101
Epoch 25/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7922 -
accuracy: 0.5280
Epoch 26/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7899 -
accuracy: 0.4686
Epoch 27/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7863 -
accuracy: 0.6349
Epoch 28/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7836 -
accuracy: 0.4919
Epoch 29/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7807 -
accuracy: 0.5566
Epoch 30/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7785 -
accuracy: 0.5253
Epoch 31/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7756 -
accuracy: 0.4817
Epoch 32/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7734 -
accuracy: 0.4284
Epoch 33/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7718 -
accuracy: 0.5493
Epoch 34/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7697 -
accuracy: 0.4517
Epoch 35/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7676 -
accuracy: 0.5574
Epoch 36/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7668 -
accuracy: 0.6501
Epoch 37/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7649 -
accuracy: 0.4866
Epoch 38/100
```

```
219/219 [==============================] - 1s 3ms/step - loss: 0.7633 -
accuracy: 0.4940
Epoch 39/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7622 -
accuracy: 0.4587
Epoch 40/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7609 -
accuracy: 0.5340
Epoch 41/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7604 -
accuracy: 0.4887
Epoch 42/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7584 -
accuracy: 0.7223
Epoch 43/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7576 -
accuracy: 0.5297
Epoch 44/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7555 -
accuracy: 0.7479
Epoch 45/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7542 -
accuracy: 0.4517
Epoch 46/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7550 -
accuracy: 0.5494
Epoch 47/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7530 -
accuracy: 0.5830
Epoch 48/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7519 -
accuracy: 0.6590
Epoch 49/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7511 -
accuracy: 0.6726
Epoch 50/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7507 -
accuracy: 0.9237
Epoch 51/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7498 -
accuracy: 0.4989
Epoch 52/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7502 -
accuracy: 0.5520
Epoch 53/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7497 -
accuracy: 0.8707
Epoch 54/100
```

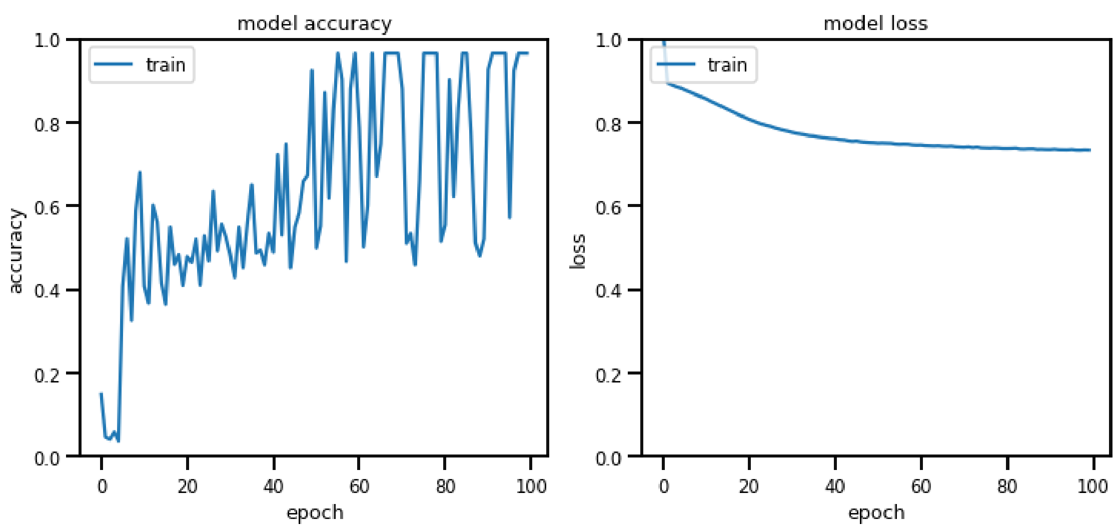
```
219/219 [==============================] - 1s 3ms/step - loss: 0.7494 -
accuracy: 0.6187
Epoch 55/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7477 -
accuracy: 0.8297
Epoch 56/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7472 -
accuracy: 0.9649
Epoch 57/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7475 -
accuracy: 0.9024
Epoch 58/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7471 -
accuracy: 0.4671
Epoch 59/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7456 -
accuracy: 0.8791
Epoch 60/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7450 -
accuracy: 0.9649
Epoch 61/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7453 -
accuracy: 0.7859
Epoch 62/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7440 -
accuracy: 0.5016
Epoch 63/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7436 -
accuracy: 0.6011
Epoch 64/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7431 -
accuracy: 0.9649
Epoch 65/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7436 -
accuracy: 0.6707
Epoch 66/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7425 -
accuracy: 0.7480
Epoch 67/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7421 -
accuracy: 0.9649
Epoch 68/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7427 -
accuracy: 0.9649
Epoch 69/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7412 -
accuracy: 0.9649
Epoch 70/100
```

```
219/219 [==============================] - 1s 4ms/step - loss: 0.7406 -
accuracy: 0.9649
Epoch 71/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7401 -
accuracy: 0.8796
Epoch 72/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7410 -
accuracy: 0.5103
Epoch 73/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7393 -
accuracy: 0.5339
Epoch 74/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7406 -
accuracy: 0.4591
Epoch 75/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7385 -
accuracy: 0.6603
Epoch 76/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7381 -
accuracy: 0.9649
Epoch 77/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7377 -
accuracy: 0.9649
Epoch 78/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7383 -
accuracy: 0.9649
Epoch 79/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7379 -
accuracy: 0.9647
Epoch 80/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7370 -
accuracy: 0.5151
Epoch 81/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7369 -
accuracy: 0.5547
Epoch 82/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7371 -
accuracy: 0.9021
Epoch 83/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7377 -
accuracy: 0.6224
Epoch 84/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7356 -
accuracy: 0.8337
Epoch 85/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7353 -
accuracy: 0.9649
Epoch 86/100
```

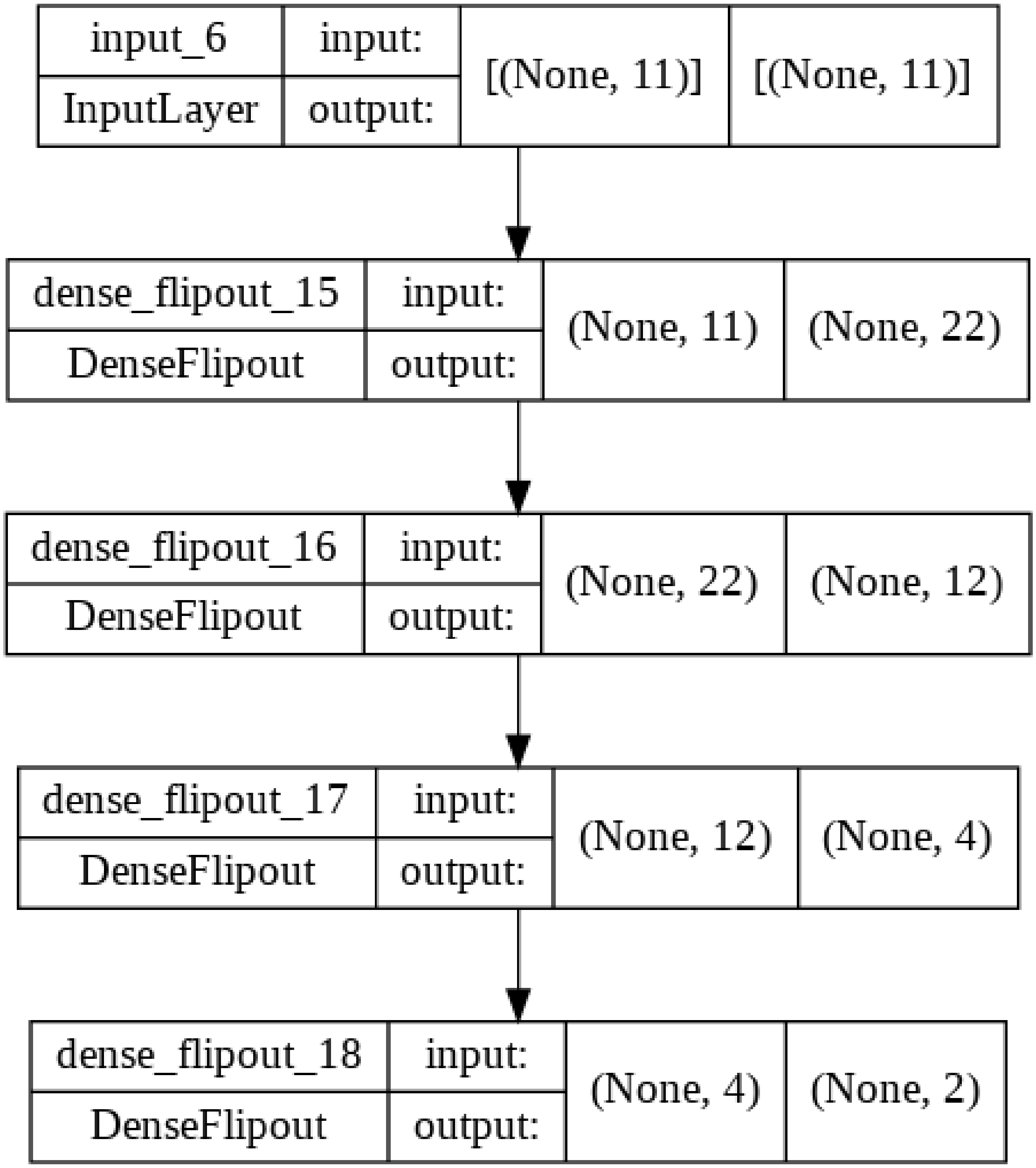
```
219/219 [==============================] - 1s 3ms/step - loss: 0.7360 -
accuracy: 0.9647
Epoch 87/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7361 -
accuracy: 0.7766
Epoch 88/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7345 -
accuracy: 0.5121
Epoch 89/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7348 -
accuracy: 0.4793
Epoch 90/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7343 -
accuracy: 0.5217
Epoch 91/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7341 -
accuracy: 0.9260
Epoch 92/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7350 -
accuracy: 0.9649
Epoch 93/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7338 -
accuracy: 0.9649
Epoch 94/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7336 -
accuracy: 0.9649
Epoch 95/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7334 -
accuracy: 0.9649
Epoch 96/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7342 -
accuracy: 0.5724
Epoch 97/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7330 -
accuracy: 0.9226
Epoch 98/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7327 -
accuracy: 0.9649
Epoch 99/100
219/219 [==============================] - 1s 4ms/step - loss: 0.7334 -
accuracy: 0.9649
Epoch 100/100
219/219 [==============================] - 1s 3ms/step - loss: 0.7331 -
accuracy: 0.9647
dict_keys(['loss', 'accuracy'])
94/94 [==============================] - 0s 3ms/step - loss: 0.4603 - accuracy:
0.9683
```

```
Test accuracy: 0.9683333039283752

Test loss: 0.46032780408859253
```

[96]:
```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'])
#plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.ylim(0, 1)
# summarize history for loss
plt.subplot(1,2,2)
plt.plot(history.history['loss'])
#plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.ylim(0, 1)
plt.show()
plot_model(normal_bnn2_model, to_file='model_plot.png', show_shapes=True,␣
↪show_layer_names=True)
```



[96]:



[97]:
```
normal_bnn2_model.save('normal_bnn2_model.h5')
normal_bnn2_model.save('saved_model/normal_bnn2_model')
```

```
INFO:tensorflow:Assets written to: saved_model/normal_bnn2_model/assets
```

2. BNN WITH DIFFERENT DROPOUT VALUES MC Dropout write description here!

[98]:
```
dist = tfp.distributions
dataset_size = len(X_train)
kl_divergence_function = (lambda q, p, _: dist.kl_divergence(q, p) / tf.
 ↪cast(dataset_size, dtype=tf.float32))

model_dropout_v1 = tf.keras.Sequential([
    tf.keras.Input(X_train.shape[1]),
    tfp.layers.DenseFlipout(14, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.2),
    tfp.layers.DenseFlipout(6, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu ),
    tfp.layers.DenseFlipout(2, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.softmax),
])

learning_rate = 0.005
model_dropout_v1.compile(optimizer=tf.keras.optimizers.
 ↪Adam(learning_rate),loss='binary_crossentropy',metrics=['accuracy'])
```

```
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:102: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:112: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
```

[99]:
```
dist = tfp.distributions
dataset_size = len(X_train)
kl_divergence_function = (lambda q, p, _: dist.kl_divergence(q, p) / tf.
 ↪cast(dataset_size, dtype=tf.float32))

model_dropout_v2 = tf.keras.Sequential([
    tf.keras.Input(X_train.shape[1]),
    tfp.layers.DenseFlipout(14, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.35),
    tfp.layers.DenseFlipout(6, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu ),
    tfp.layers.DenseFlipout(2, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.softmax),
])
```

```
learning_rate = 0.005
model_dropout_v2.compile(optimizer=tf.keras.optimizers.
 ↪Adam(learning_rate),loss='binary_crossentropy',metrics=['accuracy'])
```

```
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:102: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:112: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
```

[100]:
```
dist = tfp.distributions
dataset_size = len(X_train)
kl_divergence_function = (lambda q, p, _: dist.kl_divergence(q, p) / tf.
 ↪cast(dataset_size, dtype=tf.float32))

model_dropout_v3 = tf.keras.Sequential([
    tf.keras.Input(X_train.shape[1]),
    tfp.layers.DenseFlipout(14, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.5),
    tfp.layers.DenseFlipout(6, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu ),
    tfp.layers.DenseFlipout(2, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.softmax),
])

learning_rate = 0.005
model_dropout_v3.compile(optimizer=tf.keras.optimizers.
 ↪Adam(learning_rate),loss='binary_crossentropy',metrics=['accuracy'])
```

```
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:102: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:112: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
```

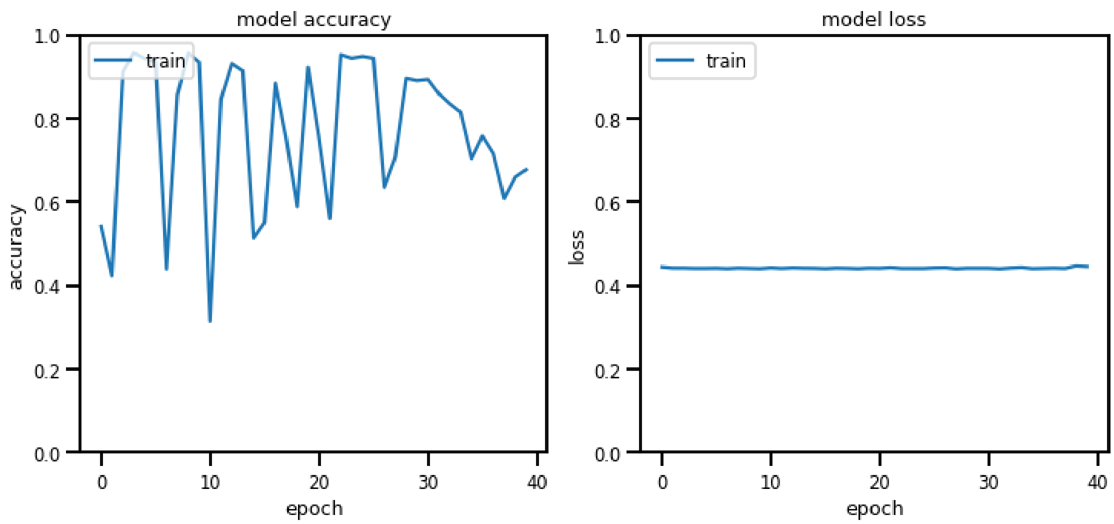
[114]:
```
from sklearn.metrics import classification_report

models = [normal_bnn_model,normal_bnn2_model,model_dropout_v1,␣
↪model_dropout_v2, model_dropout_v3]

models_acc = []
models_loss = []
i = 1
for p_model in models:
    history = p_model.fit(X_train, y_train,␣
↪epochs=40)#,batch_size=1,validation_data = (np.asarray(X_test), np.
↪asarray(y_test)),verbose=0)
    #history = normal_bnn_model.fit(np.asarray(X_train), np.
↪asarray(y_train),epochs=100, batch_size=1,validation_data = (np.
↪asarray(X_test), np.asarray(y_test)),verbose=0)
    test_loss, test_acc = p_model.evaluate(X_test, y_test)
    y_pred = p_model.predict(X_test)
    print('\nTest accuracy:', test_acc)
    print('\nTest loss:', test_loss)
    models_acc.append(test_acc)
    models_loss.append(test_loss)
    #history = normal_bnn_model.fit(np.asarray(X_train), np.
↪asarray(y_train),epochs=100, batch_size=1,verbose=0)
    # to see history:
    # list all data in history
    print(history.history.keys())
    p_model.save('%s.h5' %('p_model'+' '+str(i)))
    p_model.save('saved_model/%s' %('p_model'+' '+str(i)))
    i = i+1
    # summarize history for accuracy
    plt.figure(figsize=(12,5))
    plt.subplot(1,2,1)
    plt.plot(history.history['accuracy'])
    #plt.plot(history.history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.ylim(0, 1)
    # summarize history for loss
    plt.subplot(1,2,2)
    plt.plot(history.history['loss'])
    #plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
```

```
    plt.ylim(0, 1)
    plt.show()
    plot_model(p_model, to_file='model_plotss.png', show_shapes=True,␣
 ↪show_layer_names=True)
    '''index = 0
    for i in y_pred:
        if i<0.5:
            y_pred[index] = 0
        else:
            y_pred[index] = 1

    print(classification_report(y_test, y_pred))'''
```

```
Epoch 1/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4437 -
accuracy: 0.5416
Epoch 2/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4414 -
accuracy: 0.4229
Epoch 3/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4414 -
accuracy: 0.9109
Epoch 4/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4407 -
accuracy: 0.9563
Epoch 5/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4407 -
accuracy: 0.9427
Epoch 6/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4411 -
accuracy: 0.9351
Epoch 7/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4401 -
accuracy: 0.4393
Epoch 8/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4414 -
accuracy: 0.8557
Epoch 9/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4407 -
accuracy: 0.9559
Epoch 10/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4400 -
accuracy: 0.9341
Epoch 11/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4420 -
accuracy: 0.3140
Epoch 12/40
```

```
219/219 [==============================] - 1s 4ms/step - loss: 0.4407 -
accuracy: 0.8454
Epoch 13/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4420 -
accuracy: 0.9306
Epoch 14/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4413 -
accuracy: 0.9134
Epoch 15/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4410 -
accuracy: 0.5130
Epoch 16/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4400 -
accuracy: 0.5507
Epoch 17/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4414 -
accuracy: 0.8839
Epoch 18/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4409 -
accuracy: 0.7501
Epoch 19/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4400 -
accuracy: 0.5886
Epoch 20/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4413 -
accuracy: 0.9217
Epoch 21/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4410 -
accuracy: 0.7521
Epoch 22/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4426 -
accuracy: 0.5597
Epoch 23/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4406 -
accuracy: 0.9519
Epoch 24/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4406 -
accuracy: 0.9436
Epoch 25/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4405 -
accuracy: 0.9479
Epoch 26/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4418 -
accuracy: 0.9434
Epoch 27/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4423 -
accuracy: 0.6349
Epoch 28/40
```

```
219/219 [==============================] - 1s 4ms/step - loss: 0.4398 -
accuracy: 0.7064
Epoch 29/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4411 -
accuracy: 0.8957
Epoch 30/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4411 -
accuracy: 0.8904
Epoch 31/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4410 -
accuracy: 0.8930
Epoch 32/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4396 -
accuracy: 0.8586
Epoch 33/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4416 -
accuracy: 0.8346
Epoch 34/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4429 -
accuracy: 0.8141
Epoch 35/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4401 -
accuracy: 0.7030
Epoch 36/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4407 -
accuracy: 0.7573
Epoch 37/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4413 -
accuracy: 0.7153
Epoch 38/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4406 -
accuracy: 0.6077
Epoch 39/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4465 -
accuracy: 0.6587
Epoch 40/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4451 -
accuracy: 0.6766
94/94 [==============================] - 0s 3ms/step - loss: 0.4377 - accuracy:
0.9687

Test accuracy: 0.968666672706604

Test loss: 0.43769827485084534
dict_keys(['loss', 'accuracy'])
INFO:tensorflow:Assets written to: saved_model/p_model 1/assets
```

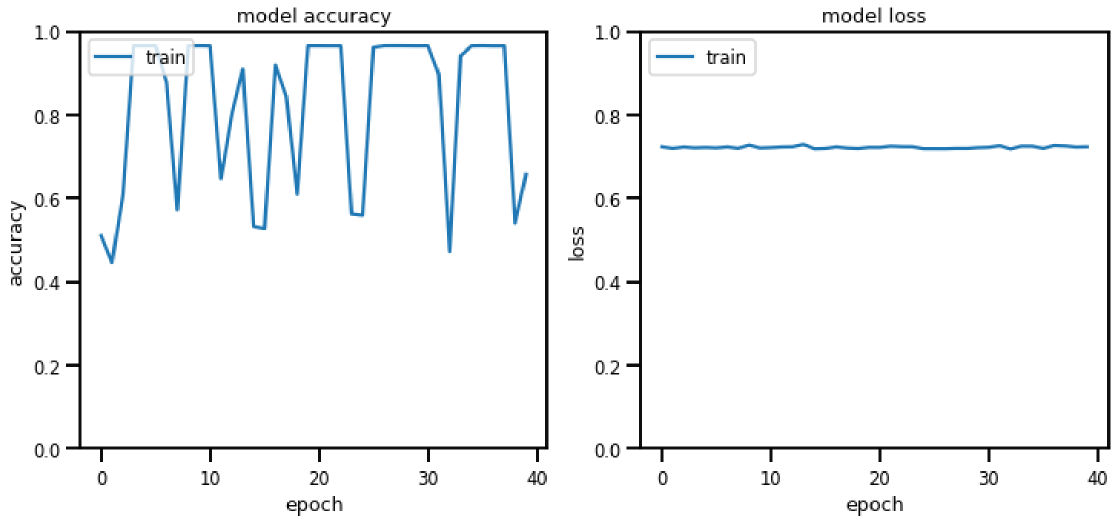



```
Epoch 1/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7227 -
accuracy: 0.5099
Epoch 2/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7189 -
accuracy: 0.4459
Epoch 3/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7222 -
accuracy: 0.6073
Epoch 4/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7202 -
accuracy: 0.9646
Epoch 5/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7212 -
accuracy: 0.9644
Epoch 6/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7201 -
accuracy: 0.9647
Epoch 7/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7225 -
accuracy: 0.8770
Epoch 8/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7190 -
accuracy: 0.5717
Epoch 9/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7269 -
accuracy: 0.9643
Epoch 10/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7199 -
accuracy: 0.9647
```

```
Epoch 11/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7208 -
accuracy: 0.9644
Epoch 12/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7223 -
accuracy: 0.6471
Epoch 13/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7227 -
accuracy: 0.8010
Epoch 14/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7286 -
accuracy: 0.9083
Epoch 15/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7177 -
accuracy: 0.5320
Epoch 16/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7186 -
accuracy: 0.5271
Epoch 17/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7225 -
accuracy: 0.9186
Epoch 18/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7197 -
accuracy: 0.8440
Epoch 19/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7185 -
accuracy: 0.6090
Epoch 20/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7213 -
accuracy: 0.9646
Epoch 21/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7212 -
accuracy: 0.9646
Epoch 22/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7241 -
accuracy: 0.9644
Epoch 23/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7230 -
accuracy: 0.9644
Epoch 24/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7228 -
accuracy: 0.5619
Epoch 25/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7181 -
accuracy: 0.5584
Epoch 26/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7180 -
accuracy: 0.9607
```

```
Epoch 27/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7179 -
accuracy: 0.9647
Epoch 28/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7188 -
accuracy: 0.9649
Epoch 29/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7187 -
accuracy: 0.9647
Epoch 30/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7206 -
accuracy: 0.9644
Epoch 31/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7215 -
accuracy: 0.9646
Epoch 32/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7252 -
accuracy: 0.8961
Epoch 33/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7173 -
accuracy: 0.4717
Epoch 34/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7242 -
accuracy: 0.9410
Epoch 35/40
219/219 [==============================] - 1s 6ms/step - loss: 0.7241 -
accuracy: 0.9644
Epoch 36/40
219/219 [==============================] - 1s 6ms/step - loss: 0.7190 -
accuracy: 0.9649
Epoch 37/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7258 -
accuracy: 0.9643
Epoch 38/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7247 -
accuracy: 0.9644
Epoch 39/40
219/219 [==============================] - 1s 6ms/step - loss: 0.7221 -
accuracy: 0.5401
Epoch 40/40
219/219 [==============================] - 1s 6ms/step - loss: 0.7227 -
accuracy: 0.6566
94/94 [==============================] - 0s 3ms/step - loss: 0.7227 - accuracy:
0.9693

Test accuracy: 0.9693333506584167

Test loss: 0.7226769924163818
```

```
dict_keys(['loss', 'accuracy'])
INFO:tensorflow:Assets written to: saved_model/p_model 2/assets
```

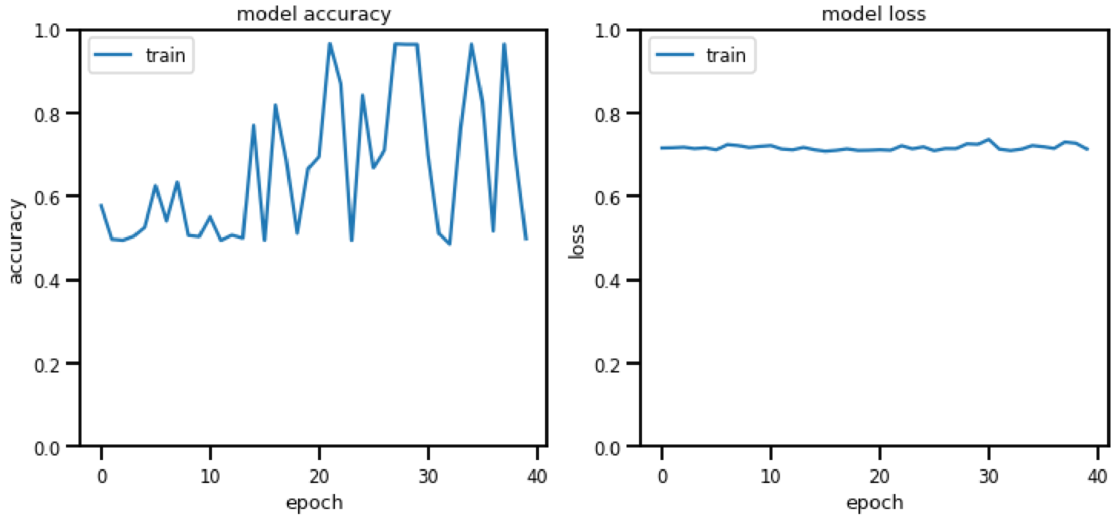



```
Epoch 1/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7149 -
accuracy: 0.5773
Epoch 2/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7154 -
accuracy: 0.4966
Epoch 3/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7168 -
accuracy: 0.4941
Epoch 4/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7134 -
accuracy: 0.5037
Epoch 5/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7154 -
accuracy: 0.5253
Epoch 6/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7105 -
accuracy: 0.6251
Epoch 7/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7233 -
accuracy: 0.5404
Epoch 8/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7208 -
accuracy: 0.6339
Epoch 9/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7160 -
accuracy: 0.5061
```

```
Epoch 10/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7188 -
accuracy: 0.5030
Epoch 11/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7208 -
accuracy: 0.5506
Epoch 12/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7124 -
accuracy: 0.4940
Epoch 13/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7103 -
accuracy: 0.5067
Epoch 14/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7163 -
accuracy: 0.4993
Epoch 15/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7106 -
accuracy: 0.7694
Epoch 16/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7070 -
accuracy: 0.4946
Epoch 17/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7091 -
accuracy: 0.8180
Epoch 18/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7127 -
accuracy: 0.6859
Epoch 19/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7089 -
accuracy: 0.5109
Epoch 20/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7093 -
accuracy: 0.6654
Epoch 21/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7105 -
accuracy: 0.6946
Epoch 22/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7094 -
accuracy: 0.9647
Epoch 23/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7204 -
accuracy: 0.8689
Epoch 24/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7129 -
accuracy: 0.4941
Epoch 25/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7177 -
accuracy: 0.8420
```

```
Epoch 26/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7083 -
accuracy: 0.6677
Epoch 27/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7136 -
accuracy: 0.7093
Epoch 28/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7134 -
accuracy: 0.9639
Epoch 29/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7251 -
accuracy: 0.9629
Epoch 30/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7239 -
accuracy: 0.9629
Epoch 31/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7359 -
accuracy: 0.7001
Epoch 32/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7119 -
accuracy: 0.5104
Epoch 33/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7085 -
accuracy: 0.4850
Epoch 34/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7121 -
accuracy: 0.7640
Epoch 35/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7208 -
accuracy: 0.9636
Epoch 36/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7179 -
accuracy: 0.8259
Epoch 37/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7138 -
accuracy: 0.5161
Epoch 38/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7295 -
accuracy: 0.9636
Epoch 39/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7266 -
accuracy: 0.7030
Epoch 40/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7126 -
accuracy: 0.4980
94/94 [==============================] - 0s 2ms/step - loss: 0.7029 - accuracy:
0.9690
```

```
Test accuracy: 0.968999981880188

Test loss: 0.7028810977935791
dict_keys(['loss', 'accuracy'])
INFO:tensorflow:Assets written to: saved_model/p_model 3/assets
```

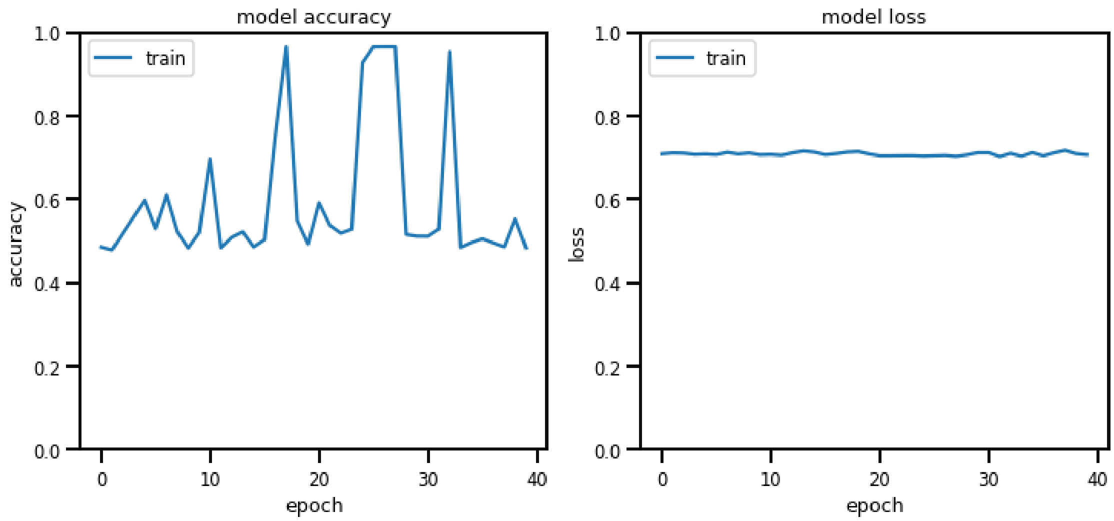



```
Epoch 1/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7085 -
accuracy: 0.4849
Epoch 2/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7107 -
accuracy: 0.4779
Epoch 3/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7101 -
accuracy: 0.5181
Epoch 4/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7071 -
accuracy: 0.5574
Epoch 5/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7080 -
accuracy: 0.5976
Epoch 6/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7065 -
accuracy: 0.5294
Epoch 7/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7121 -
accuracy: 0.6101
Epoch 8/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7080 -
accuracy: 0.5214
```

```
Epoch 9/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7106 -
accuracy: 0.4824
Epoch 10/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7063 -
accuracy: 0.5196
Epoch 11/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7070 -
accuracy: 0.6964
Epoch 12/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7052 -
accuracy: 0.4833
Epoch 13/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7110 -
accuracy: 0.5087
Epoch 14/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7153 -
accuracy: 0.5216
Epoch 15/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7126 -
accuracy: 0.4850
Epoch 16/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7064 -
accuracy: 0.5027
Epoch 17/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7090 -
accuracy: 0.7533
Epoch 18/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7127 -
accuracy: 0.9647
Epoch 19/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7141 -
accuracy: 0.5486
Epoch 20/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7083 -
accuracy: 0.4921
Epoch 21/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7037 -
accuracy: 0.5914
Epoch 22/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7038 -
accuracy: 0.5371
Epoch 23/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7041 -
accuracy: 0.5184
Epoch 24/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7043 -
accuracy: 0.5279
```

```
Epoch 25/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7031 -
accuracy: 0.9269
Epoch 26/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7039 -
accuracy: 0.9647
Epoch 27/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7047 -
accuracy: 0.9649
Epoch 28/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7025 -
accuracy: 0.9649
Epoch 29/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7060 -
accuracy: 0.5154
Epoch 30/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7112 -
accuracy: 0.5114
Epoch 31/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7112 -
accuracy: 0.5111
Epoch 32/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7020 -
accuracy: 0.5283
Epoch 33/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7095 -
accuracy: 0.9527
Epoch 34/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7029 -
accuracy: 0.4841
Epoch 35/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7114 -
accuracy: 0.4959
Epoch 36/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7039 -
accuracy: 0.5049
Epoch 37/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7109 -
accuracy: 0.4949
Epoch 38/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7167 -
accuracy: 0.4854
Epoch 39/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7088 -
accuracy: 0.5527
Epoch 40/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7064 -
accuracy: 0.4830
```

```
94/94 [==============================] - 0s 3ms/step - loss: 0.7039 - accuracy:
0.0313

Test accuracy: 0.03133333474397659

Test loss: 0.703898012638092
dict_keys(['loss', 'accuracy'])
INFO:tensorflow:Assets written to: saved_model/p_model 4/assets
```

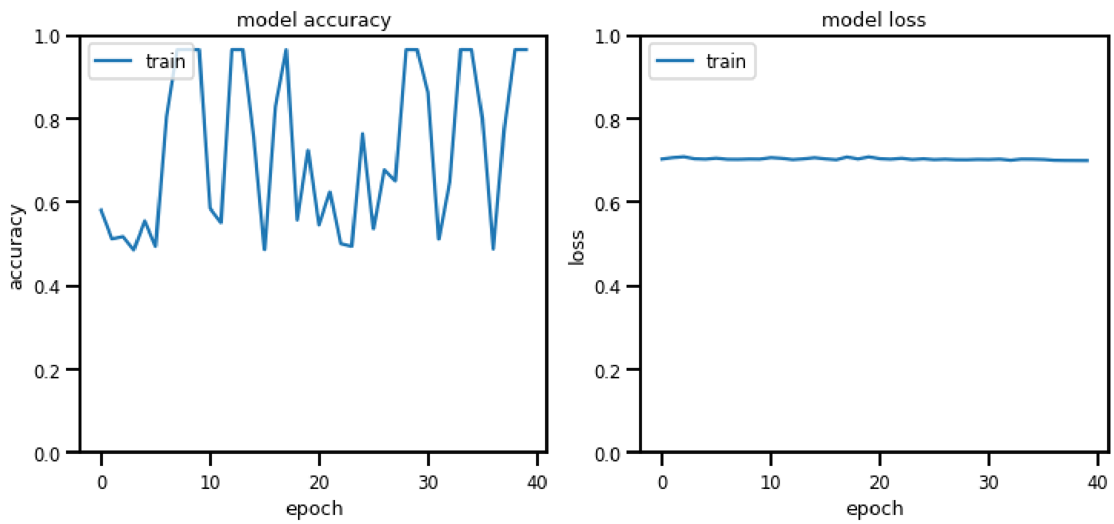



```
Epoch 1/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7029 -
accuracy: 0.5803
Epoch 2/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7063 -
accuracy: 0.5121
Epoch 3/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7085 -
accuracy: 0.5177
Epoch 4/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7035 -
accuracy: 0.4850
Epoch 5/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7026 -
accuracy: 0.5550
Epoch 6/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7049 -
accuracy: 0.4936
Epoch 7/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7022 -
accuracy: 0.8043
```

```
Epoch 8/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7020 -
accuracy: 0.9649
Epoch 9/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7028 -
accuracy: 0.9650
Epoch 10/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7026 -
accuracy: 0.9647
Epoch 11/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7063 -
accuracy: 0.5851
Epoch 12/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7046 -
accuracy: 0.5507
Epoch 13/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7015 -
accuracy: 0.9649
Epoch 14/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7033 -
accuracy: 0.9649
Epoch 15/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7061 -
accuracy: 0.7563
Epoch 16/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7033 -
accuracy: 0.4860
Epoch 17/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7011 -
accuracy: 0.8291
Epoch 18/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7079 -
accuracy: 0.9646
Epoch 19/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7030 -
accuracy: 0.5571
Epoch 20/40
219/219 [==============================] - 1s 3ms/step - loss: 0.7082 -
accuracy: 0.7241
Epoch 21/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7037 -
accuracy: 0.5449
Epoch 22/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7025 -
accuracy: 0.6244
Epoch 23/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7047 -
accuracy: 0.5004
```

```
Epoch 24/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7017 -
accuracy: 0.4939
Epoch 25/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7035 -
accuracy: 0.7637
Epoch 26/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7014 -
accuracy: 0.5361
Epoch 27/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7024 -
accuracy: 0.6777
Epoch 28/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7012 -
accuracy: 0.6507
Epoch 29/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7011 -
accuracy: 0.9649
Epoch 30/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7019 -
accuracy: 0.9647
Epoch 31/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7016 -
accuracy: 0.8624
Epoch 32/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7026 -
accuracy: 0.5119
Epoch 33/40
219/219 [==============================] - 1s 4ms/step - loss: 0.6998 -
accuracy: 0.6477
Epoch 34/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7027 -
accuracy: 0.9649
Epoch 35/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7025 -
accuracy: 0.9647
Epoch 36/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7018 -
accuracy: 0.7990
Epoch 37/40
219/219 [==============================] - 1s 4ms/step - loss: 0.6998 -
accuracy: 0.4873
Epoch 38/40
219/219 [==============================] - 1s 4ms/step - loss: 0.6994 -
accuracy: 0.7760
Epoch 39/40
219/219 [==============================] - 1s 4ms/step - loss: 0.6993 -
accuracy: 0.9649
```

```
Epoch 40/40
219/219 [==============================] - 1s 4ms/step - loss: 0.6991 -
accuracy: 0.9649
94/94 [==============================] - 0s 2ms/step - loss: 0.7060 - accuracy:
0.9687

Test accuracy: 0.968666672706604

Test loss: 0.7059929370880127
dict_keys(['loss', 'accuracy'])
INFO:tensorflow:Assets written to: saved_model/p_model 5/assets
```



[115]:
```
print(models_acc)
print(models_loss)
```

```
[0.968666672706604, 0.9693333506584167, 0.968999981880188, 0.03133333474397659,
0.968666672706604]
[0.43769827485084534, 0.7226769924163818, 0.7028810977935791, 0.703898012638092,
0.7059929370880127]
```

3. BNN WITH DIFFERENT EARLY STOPS

[116]:
```
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)

#callbacks=[callback]
dist = tfp.distributions
dataset_size = len(X_train)
kl_divergence_function = (lambda q, p, _: dist.kl_divergence(q, p) / tf.
 ↪cast(dataset_size, dtype=tf.float32))
```

```
model_callback_v1 = tf.keras.Sequential([
    tf.keras.Input(X_train.shape[1]),
    tfp.layers.DenseFlipout(14, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu),
    #tf.keras.layers.Dropout(0.5)
    tfp.layers.DenseFlipout(6, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu ),
    tfp.layers.DenseFlipout(2, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.softmax),
])

learning_rate = 0.005
model_callback_v1.compile(optimizer=tf.keras.optimizers.
 ↪Adam(learning_rate),loss='binary_crossentropy',metrics=['accuracy'])
history = model_callback_v1.fit(np.asarray(X_train), np.
 ↪asarray(y_train),epochs=80, batch_size=1, callbacks=[callback],verbose=0)
len(history.history['loss'])

#model_tfp_v2.fit(X_train, y_train, epochs=80)
test_loss, test_acc = model_callback_v1.evaluate(X_test,  y_test)
print('\nTest accuracy:', test_acc)
print('\nTest loss:', test_loss)

model_callback_v1.summary()
```

```
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:102: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:112: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
```

```
94/94 [==============================] - 1s 2ms/step - loss: 0.8611 - accuracy:
0.9593

Test accuracy: 0.9593333601951599

Test loss: 0.861093282699585
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_flipout_37 (DenseFlip  (None, 14)               322
```

```
 out)

 dense_flipout_38 (DenseFlip  (None, 6)              174
 out)

 dense_flipout_39 (DenseFlip  (None, 2)              26
 out)

=================================================================
Total params: 522
Trainable params: 522
Non-trainable params: 0
_________________________________________________________________
```

[117]:
```
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=4)

#callbacks=[callback]
dist = tfp.distributions
dataset_size = len(X_train)
kl_divergence_function = (lambda q, p, _: dist.kl_divergence(q, p) / tf.
 ↪cast(dataset_size, dtype=tf.float32))

model_callback_v2 = tf.keras.Sequential([
    tf.keras.Input(X_train.shape[1]),
    tfp.layers.DenseFlipout(14, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu),
    #tf.keras.layers.Dropout(0.5)
    tfp.layers.DenseFlipout(6, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu ),
    tfp.layers.DenseFlipout(2, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.softmax),
])

learning_rate = 0.005
model_callback_v2.compile(optimizer=tf.keras.optimizers.
 ↪Adam(learning_rate),loss='binary_crossentropy',metrics=['accuracy'])
history = model_callback_v2.fit(np.asarray(X_train), np.
 ↪asarray(y_train),epochs=10, batch_size=1, callbacks=[callback],verbose=0)
len(history.history['loss'])

test_loss, test_acc = model_callback_v2.evaluate(X_test,  y_test)
print('\nTest accuracy:', test_acc)
print('\nTest loss:', test_loss)
model_callback_v2.summary()
```

```
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:102: UserWarning:
```

```
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:112: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)

94/94 [==============================] - 1s 2ms/step - loss: 0.7917 - accuracy:
0.9607

Test accuracy: 0.9606666564941406

Test loss: 0.7917195558547974
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_flipout_40 (DenseFlip  (None, 14)               322
 out)

 dense_flipout_41 (DenseFlip  (None, 6)                174
 out)

 dense_flipout_42 (DenseFlip  (None, 2)                26
 out)

=================================================================
Total params: 522
Trainable params: 522
Non-trainable params: 0
_________________________________________________________________
```

[118]:

```
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=2)

#callbacks=[callback]
dist = tfp.distributions
dataset_size = len(X_train)
kl_divergence_function = (lambda q, p, _: dist.kl_divergence(q, p) / tf.
 ↪cast(dataset_size, dtype=tf.float32))

model_callback_v3 = tf.keras.Sequential([
    tf.keras.Input(X_train.shape[1]),
    tfp.layers.DenseFlipout(14, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu),
    #tf.keras.layers.Dropout(0.5)
```

```
    tfp.layers.DenseFlipout(6, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.relu ),
    tfp.layers.DenseFlipout(2, kernel_divergence_fn=kl_divergence_function,␣
 ↪activation=tf.nn.softmax),
])

learning_rate = 0.005
model_callback_v3.compile(optimizer=tf.keras.optimizers.
 ↪Adam(learning_rate),loss='binary_crossentropy',metrics=['accuracy'])
history = model_callback_v3.fit(np.asarray(X_train), np.
 ↪asarray(y_train),epochs=10, batch_size=1, callbacks=[callback],verbose=0)
len(history.history['loss'])

test_loss, test_acc = model_callback_v3.evaluate(X_test,  y_test)
print('\nTest accuracy:', test_acc)
print('\nTest loss:', test_loss)
model_callback_v3.summary()
```

```
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:102: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
/usr/local/lib/python3.7/dist-
packages/tensorflow_probability/python/layers/util.py:112: UserWarning:
`layer.add_variable` is deprecated and will be removed in a future version.
Please use `layer.add_weight` method instead.
  trainable=trainable)
```

```
94/94 [==============================] - 1s 2ms/step - loss: 0.9403 - accuracy:
0.9460

Test accuracy: 0.9459999799728394

Test loss: 0.9402525424957275
Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_flipout_43 (DenseFlip  (None, 14)               322
 out)

 dense_flipout_44 (DenseFlip  (None, 6)                174
 out)

 dense_flipout_45 (DenseFlip  (None, 2)                26
 out)
```

```
=================================================================
Total params: 522
Trainable params: 522
Non-trainable params: 0
_________________________________________________________________
```

[119]:

```
from sklearn.metrics import classification_report

models = [normal_bnn_model,normal_bnn2_model,model_callback_v1,␣
 ↪model_callback_v2, model_callback_v3]

models_acc = []
models_loss = []
i = 6
for p_model in models:
    history = p_model.fit(X_train, y_train,␣
 ↪epochs=40)#,batch_size=1,validation_data = (np.asarray(X_test), np.
 ↪asarray(y_test)),verbose=0)
    #history = normal_bnn_model.fit(np.asarray(X_train), np.
 ↪asarray(y_train),epochs=100, batch_size=1,validation_data = (np.
 ↪asarray(X_test), np.asarray(y_test)),verbose=0)
    test_loss, test_acc = p_model.evaluate(X_test, y_test)
    y_pred = p_model.predict(X_test)
    print('\nTest accuracy:', test_acc)
    print('\nTest loss:', test_loss)
    models_acc.append(test_acc)
    models_loss.append(test_loss)
    #history = normal_bnn_model.fit(np.asarray(X_train), np.
 ↪asarray(y_train),epochs=100, batch_size=1,verbose=0)
    # to see history:
    # list all data in history
    print(history.history.keys())
    p_model.save('%s.h5' %('callp_model'+' '+str(i)))
    p_model.save('saved_model/%s' %('callp_model'+' '+str(i)))
    i = i+1
    # summarize history for accuracy
    plt.figure(figsize=(12,5))
    plt.subplot(1,2,1)
    plt.plot(history.history['accuracy'])
    #plt.plot(history.history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.ylim(0, 1)
    # summarize history for loss
    plt.subplot(1,2,2)
```

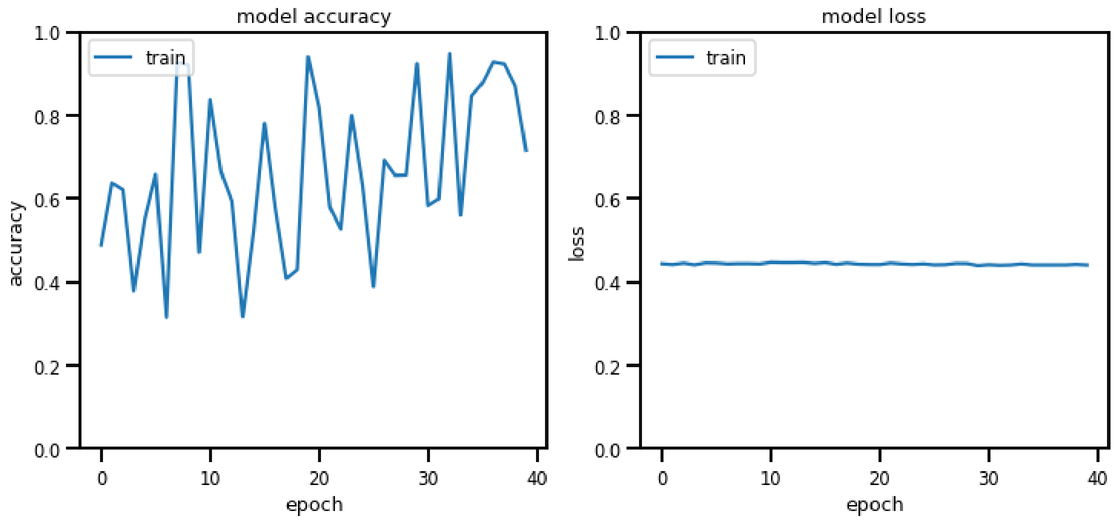
```python
    plt.plot(history.history['loss'])
    #plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.ylim(0, 1)
    plt.show()
    plot_model(p_model, to_file='model_plotsss.png', show_shapes=True,␣
↪show_layer_names=True)
    '''index = 0
    for i in y_pred:
        if i<0.5:
            y_pred[index] = 0
        else:
            y_pred[index] = 1

    print(classification_report(y_test, y_pred))'''
```

```
Epoch 1/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4434 -
accuracy: 0.4884
Epoch 2/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4416 -
accuracy: 0.6369
Epoch 3/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4449 -
accuracy: 0.6207
Epoch 4/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4409 -
accuracy: 0.3781
Epoch 5/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4455 -
accuracy: 0.5490
Epoch 6/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4451 -
accuracy: 0.6577
Epoch 7/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4430 -
accuracy: 0.3147
Epoch 8/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4436 -
accuracy: 0.9260
Epoch 9/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4436 -
accuracy: 0.9197
Epoch 10/40
```

```
219/219 [==============================] - 1s 4ms/step - loss: 0.4429 -
accuracy: 0.4706
Epoch 11/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4469 -
accuracy: 0.8370
Epoch 12/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4463 -
accuracy: 0.6643
Epoch 13/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4463 -
accuracy: 0.5936
Epoch 14/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4467 -
accuracy: 0.3163
Epoch 15/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4443 -
accuracy: 0.5209
Epoch 16/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4463 -
accuracy: 0.7796
Epoch 17/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4421 -
accuracy: 0.5740
Epoch 18/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4451 -
accuracy: 0.4077
Epoch 19/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4424 -
accuracy: 0.4290
Epoch 20/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4418 -
accuracy: 0.9400
Epoch 21/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4416 -
accuracy: 0.8180
Epoch 22/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4451 -
accuracy: 0.5791
Epoch 23/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4430 -
accuracy: 0.5261
Epoch 24/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4418 -
accuracy: 0.7989
Epoch 25/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4432 -
accuracy: 0.6311
Epoch 26/40
```

```
219/219 [==============================] - 1s 4ms/step - loss: 0.4408 -
accuracy: 0.3891
Epoch 27/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4412 -
accuracy: 0.6920
Epoch 28/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4441 -
accuracy: 0.6551
Epoch 29/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4439 -
accuracy: 0.6560
Epoch 30/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4392 -
accuracy: 0.9227
Epoch 31/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4412 -
accuracy: 0.5830
Epoch 32/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4399 -
accuracy: 0.5991
Epoch 33/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4406 -
accuracy: 0.9476
Epoch 34/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4430 -
accuracy: 0.5597
Epoch 35/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4407 -
accuracy: 0.8460
Epoch 36/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4406 -
accuracy: 0.8760
Epoch 37/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4406 -
accuracy: 0.9269
Epoch 38/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4406 -
accuracy: 0.9221
Epoch 39/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4419 -
accuracy: 0.8694
Epoch 40/40
219/219 [==============================] - 1s 4ms/step - loss: 0.4406 -
accuracy: 0.7147
94/94 [==============================] - 0s 4ms/step - loss: 0.4379 - accuracy:
0.9687

Test accuracy: 0.968666672706604
```

```
Test loss: 0.43793368339538574
dict_keys(['loss', 'accuracy'])
INFO:tensorflow:Assets written to: saved_model/callp_model 6/assets
```

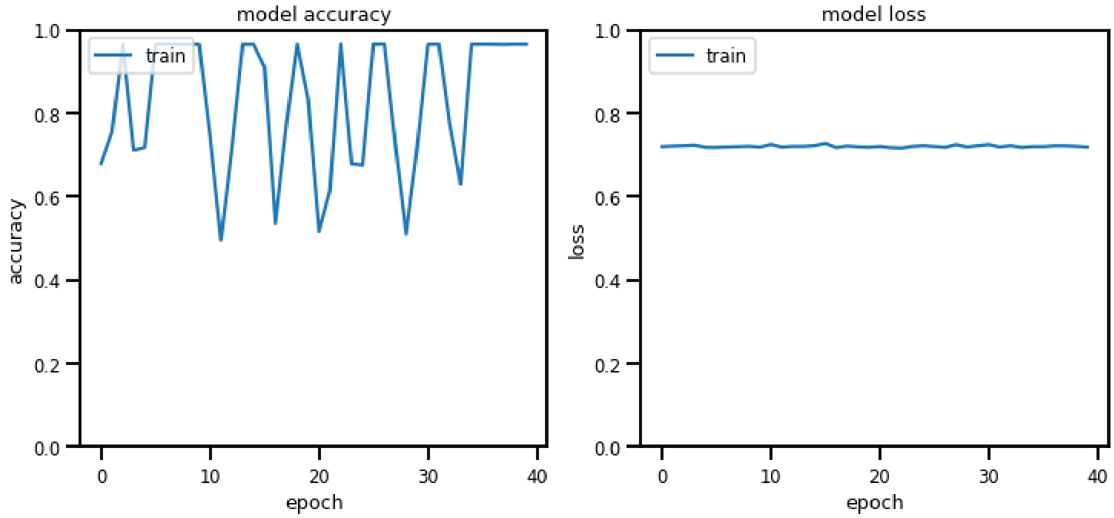



```
Epoch 1/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7195 -
accuracy: 0.6791
Epoch 2/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7206 -
accuracy: 0.7540
Epoch 3/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7216 -
accuracy: 0.9643
Epoch 4/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7229 -
accuracy: 0.7107
Epoch 5/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7175 -
accuracy: 0.7173
Epoch 6/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7175 -
accuracy: 0.9647
Epoch 7/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7184 -
accuracy: 0.9647
Epoch 8/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7193 -
accuracy: 0.9644
Epoch 9/40
```

```
219/219 [==============================] - 1s 5ms/step - loss: 0.7202 -
accuracy: 0.9643
Epoch 10/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7181 -
accuracy: 0.9646
Epoch 11/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7248 -
accuracy: 0.7440
Epoch 12/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7182 -
accuracy: 0.4951
Epoch 13/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7200 -
accuracy: 0.7143
Epoch 14/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7199 -
accuracy: 0.9643
Epoch 15/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7218 -
accuracy: 0.9643
Epoch 16/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7271 -
accuracy: 0.9097
Epoch 17/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7168 -
accuracy: 0.5347
Epoch 18/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7208 -
accuracy: 0.7706
Epoch 19/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7187 -
accuracy: 0.9647
Epoch 20/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7178 -
accuracy: 0.8347
Epoch 21/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7197 -
accuracy: 0.5163
Epoch 22/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7168 -
accuracy: 0.6131
Epoch 23/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7157 -
accuracy: 0.9649
Epoch 24/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7198 -
accuracy: 0.6786
Epoch 25/40
```

```
219/219 [==============================] - 1s 5ms/step - loss: 0.7216 -
accuracy: 0.6753
Epoch 26/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7196 -
accuracy: 0.9646
Epoch 27/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7176 -
accuracy: 0.9650
Epoch 28/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7242 -
accuracy: 0.7270
Epoch 29/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7184 -
accuracy: 0.5100
Epoch 30/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7215 -
accuracy: 0.7149
Epoch 31/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7245 -
accuracy: 0.9643
Epoch 32/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7185 -
accuracy: 0.9646
Epoch 33/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7216 -
accuracy: 0.7734
Epoch 34/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7174 -
accuracy: 0.6301
Epoch 35/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7194 -
accuracy: 0.9643
Epoch 36/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7193 -
accuracy: 0.9647
Epoch 37/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7213 -
accuracy: 0.9644
Epoch 38/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7212 -
accuracy: 0.9640
Epoch 39/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7202 -
accuracy: 0.9647
Epoch 40/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7181 -
accuracy: 0.9644
94/94 [==============================] - 0s 3ms/step - loss: 0.7183 - accuracy:
```

```
0.9690

Test accuracy: 0.968999981880188

Test loss: 0.7182931303977966
dict_keys(['loss', 'accuracy'])
INFO:tensorflow:Assets written to: saved_model/callp_model 7/assets
```

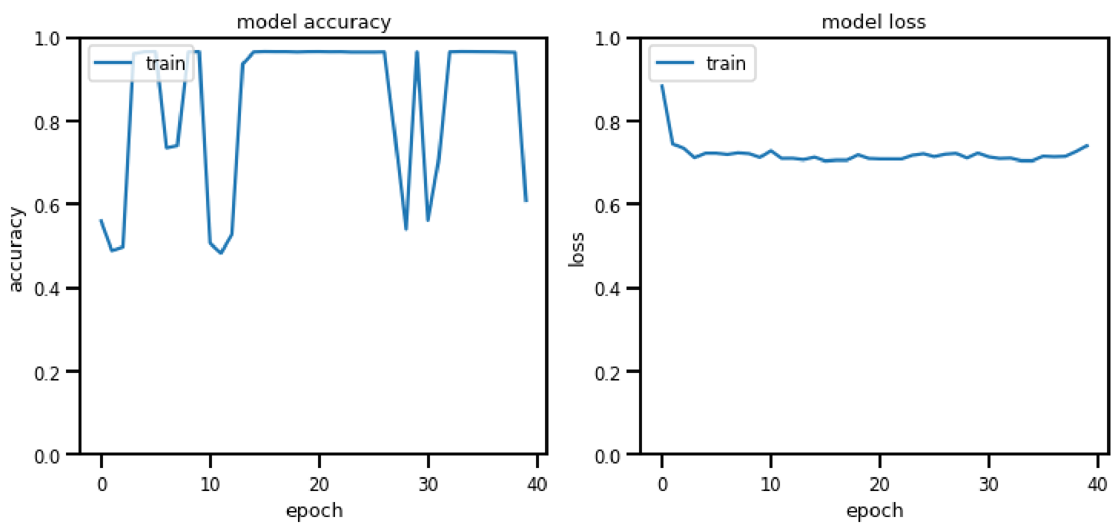



```
Epoch 1/40
219/219 [==============================] - 2s 4ms/step - loss: 0.8826 -
accuracy: 0.5593
Epoch 2/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7445 -
accuracy: 0.4886
Epoch 3/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7337 -
accuracy: 0.4974
Epoch 4/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7109 -
accuracy: 0.9599
Epoch 5/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7218 -
accuracy: 0.9644
Epoch 6/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7217 -
accuracy: 0.9647
Epoch 7/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7187 -
accuracy: 0.7347
Epoch 8/40
```

```
219/219 [==============================] - 1s 4ms/step - loss: 0.7227 -
accuracy: 0.7401
Epoch 9/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7206 -
accuracy: 0.9639
Epoch 10/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7117 -
accuracy: 0.9647
Epoch 11/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7278 -
accuracy: 0.5063
Epoch 12/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7092 -
accuracy: 0.4826
Epoch 13/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7094 -
accuracy: 0.5276
Epoch 14/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7064 -
accuracy: 0.9357
Epoch 15/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7123 -
accuracy: 0.9640
Epoch 16/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7033 -
accuracy: 0.9649
Epoch 17/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7052 -
accuracy: 0.9646
Epoch 18/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7052 -
accuracy: 0.9646
Epoch 19/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7180 -
accuracy: 0.9639
Epoch 20/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7090 -
accuracy: 0.9646
Epoch 21/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7080 -
accuracy: 0.9647
Epoch 22/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7079 -
accuracy: 0.9643
Epoch 23/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7078 -
accuracy: 0.9644
Epoch 24/40
```

```
219/219 [==============================] - 1s 4ms/step - loss: 0.7167 -
accuracy: 0.9636
Epoch 25/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7206 -
accuracy: 0.9636
Epoch 26/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7135 -
accuracy: 0.9636
Epoch 27/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7194 -
accuracy: 0.9640
Epoch 28/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7214 -
accuracy: 0.7577
Epoch 29/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7102 -
accuracy: 0.5403
Epoch 30/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7220 -
accuracy: 0.9637
Epoch 31/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7129 -
accuracy: 0.5607
Epoch 32/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7090 -
accuracy: 0.7081
Epoch 33/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7099 -
accuracy: 0.9641
Epoch 34/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7038 -
accuracy: 0.9649
Epoch 35/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7036 -
accuracy: 0.9647
Epoch 36/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7144 -
accuracy: 0.9644
Epoch 37/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7132 -
accuracy: 0.9643
Epoch 38/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7141 -
accuracy: 0.9637
Epoch 39/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7258 -
accuracy: 0.9630
Epoch 40/40
```

```
219/219 [==============================] - 1s 4ms/step - loss: 0.7398 -
accuracy: 0.6083
94/94 [==============================] - 0s 2ms/step - loss: 0.7300 - accuracy:
0.9673

Test accuracy: 0.9673333168029785

Test loss: 0.7300313115119934
dict_keys(['loss', 'accuracy'])
INFO:tensorflow:Assets written to: saved_model/callp_model 8/assets
```

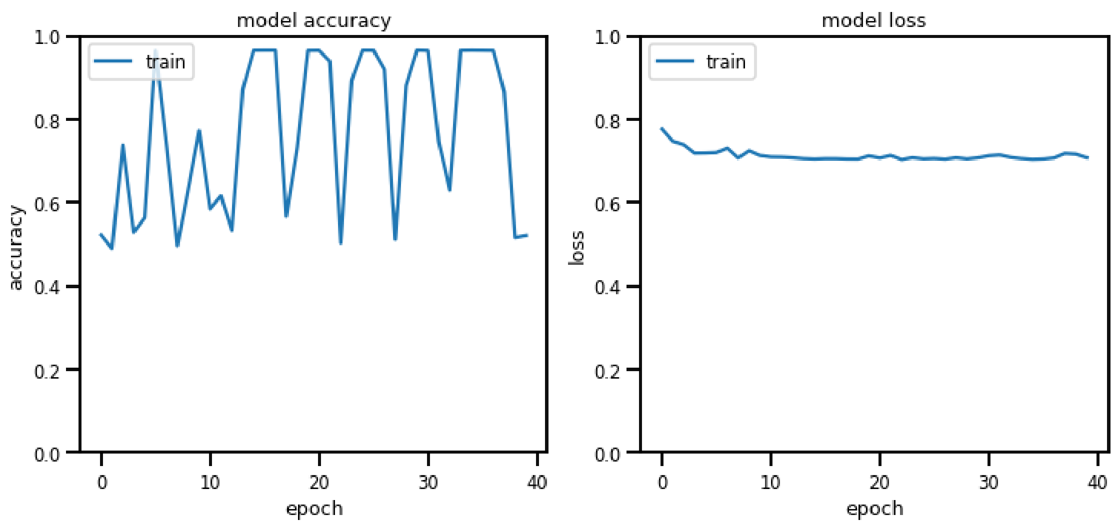



```
Epoch 1/40
219/219 [==============================] - 2s 4ms/step - loss: 0.7760 -
accuracy: 0.5219
Epoch 2/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7468 -
accuracy: 0.4893
Epoch 3/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7383 -
accuracy: 0.7376
Epoch 4/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7180 -
accuracy: 0.5283
Epoch 5/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7184 -
accuracy: 0.5636
Epoch 6/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7193 -
accuracy: 0.9640
Epoch 7/40
```

```
219/219 [==============================] - 1s 4ms/step - loss: 0.7300 -
accuracy: 0.7379
Epoch 8/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7066 -
accuracy: 0.4960
Epoch 9/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7238 -
accuracy: 0.6319
Epoch 10/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7125 -
accuracy: 0.7720
Epoch 11/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7092 -
accuracy: 0.5844
Epoch 12/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7087 -
accuracy: 0.6157
Epoch 13/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7073 -
accuracy: 0.5327
Epoch 14/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7050 -
accuracy: 0.8707
Epoch 15/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7039 -
accuracy: 0.9647
Epoch 16/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7048 -
accuracy: 0.9649
Epoch 17/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7048 -
accuracy: 0.9649
Epoch 18/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7039 -
accuracy: 0.5666
Epoch 19/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7037 -
accuracy: 0.7311
Epoch 20/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7116 -
accuracy: 0.9643
Epoch 21/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7065 -
accuracy: 0.9646
Epoch 22/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7126 -
accuracy: 0.9383
Epoch 23/40
```

```
219/219 [==============================] - 1s 4ms/step - loss: 0.7025 -
accuracy: 0.5023
Epoch 24/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7074 -
accuracy: 0.8920
Epoch 25/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7044 -
accuracy: 0.9647
Epoch 26/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7053 -
accuracy: 0.9646
Epoch 27/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7035 -
accuracy: 0.9189
Epoch 28/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7073 -
accuracy: 0.5114
Epoch 29/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7042 -
accuracy: 0.8801
Epoch 30/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7071 -
accuracy: 0.9650
Epoch 31/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7119 -
accuracy: 0.9641
Epoch 32/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7137 -
accuracy: 0.7447
Epoch 33/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7082 -
accuracy: 0.6293
Epoch 34/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7052 -
accuracy: 0.9644
Epoch 35/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7031 -
accuracy: 0.9649
Epoch 36/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7040 -
accuracy: 0.9646
Epoch 37/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7068 -
accuracy: 0.9643
Epoch 38/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7174 -
accuracy: 0.8647
Epoch 39/40
```

```
219/219 [==============================] - 1s 4ms/step - loss: 0.7158 -
accuracy: 0.5154
Epoch 40/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7073 -
accuracy: 0.5206
94/94 [==============================] - 0s 3ms/step - loss: 0.7086 - accuracy:
0.9680

Test accuracy: 0.9679999947547913

Test loss: 0.708635687828064
dict_keys(['loss', 'accuracy'])
INFO:tensorflow:Assets written to: saved_model/callp_model 9/assets
```

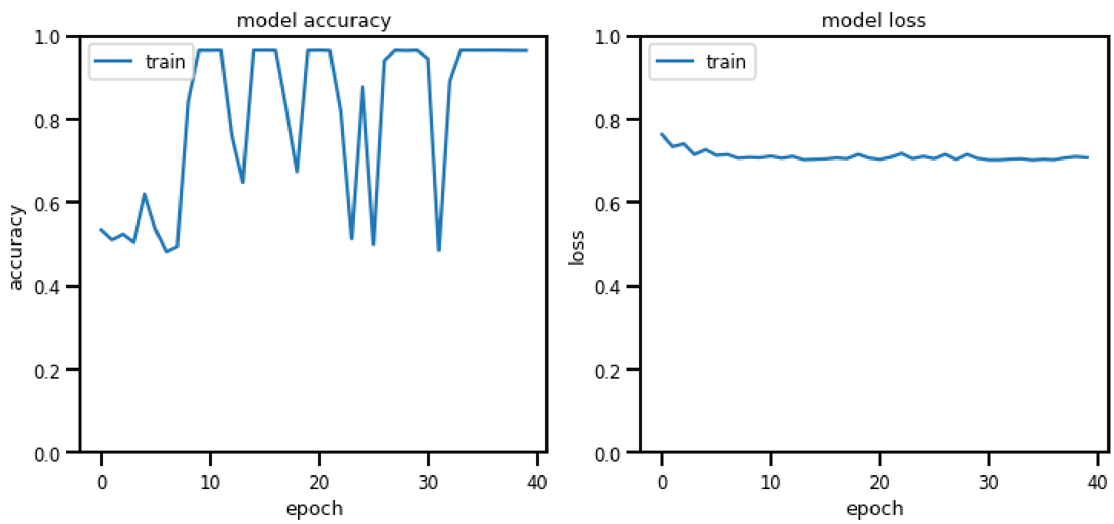



```
Epoch 1/40
219/219 [==============================] - 2s 4ms/step - loss: 0.7626 -
accuracy: 0.5341
Epoch 2/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7341 -
accuracy: 0.5100
Epoch 3/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7411 -
accuracy: 0.5236
Epoch 4/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7150 -
accuracy: 0.5039
Epoch 5/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7271 -
accuracy: 0.6199
Epoch 6/40
```

```
219/219 [==============================] - 1s 4ms/step - loss: 0.7132 -
accuracy: 0.5349
Epoch 7/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7154 -
accuracy: 0.4819
Epoch 8/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7064 -
accuracy: 0.4943
Epoch 9/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7084 -
accuracy: 0.8410
Epoch 10/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7073 -
accuracy: 0.9644
Epoch 11/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7113 -
accuracy: 0.9643
Epoch 12/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7063 -
accuracy: 0.9647
Epoch 13/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7106 -
accuracy: 0.7609
Epoch 14/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7023 -
accuracy: 0.6486
Epoch 15/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7033 -
accuracy: 0.9647
Epoch 16/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7042 -
accuracy: 0.9649
Epoch 17/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7071 -
accuracy: 0.9644
Epoch 18/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7050 -
accuracy: 0.8233
Epoch 19/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7158 -
accuracy: 0.6729
Epoch 20/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7068 -
accuracy: 0.9643
Epoch 21/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7028 -
accuracy: 0.9649
Epoch 22/40
```

```
219/219 [==============================] - 1s 5ms/step - loss: 0.7086 -
accuracy: 0.9641
Epoch 23/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7178 -
accuracy: 0.8193
Epoch 24/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7051 -
accuracy: 0.5131
Epoch 25/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7105 -
accuracy: 0.8766
Epoch 26/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7050 -
accuracy: 0.4999
Epoch 27/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7158 -
accuracy: 0.9397
Epoch 28/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7029 -
accuracy: 0.9647
Epoch 29/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7157 -
accuracy: 0.9639
Epoch 30/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7057 -
accuracy: 0.9647
Epoch 31/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7017 -
accuracy: 0.9440
Epoch 32/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7017 -
accuracy: 0.4854
Epoch 33/40
219/219 [==============================] - 1s 4ms/step - loss: 0.7036 -
accuracy: 0.8909
Epoch 34/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7045 -
accuracy: 0.9647
Epoch 35/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7014 -
accuracy: 0.9649
Epoch 36/40
219/219 [==============================] - 1s 6ms/step - loss: 0.7032 -
accuracy: 0.9646
Epoch 37/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7021 -
accuracy: 0.9649
Epoch 38/40
```

```
219/219 [==============================] - 1s 5ms/step - loss: 0.7069 -
accuracy: 0.9644
Epoch 39/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7098 -
accuracy: 0.9641
Epoch 40/40
219/219 [==============================] - 1s 5ms/step - loss: 0.7076 -
accuracy: 0.9640
94/94 [==============================] - 0s 3ms/step - loss: 0.7239 - accuracy:
0.9657

Test accuracy: 0.965666651725769

Test loss: 0.7239024043083191
dict_keys(['loss', 'accuracy'])
INFO:tensorflow:Assets written to: saved_model/callp_model 10/assets
```



4. BNN WITH DIFFERENT REGULARIZERS `TRANSFORMERS`

6. MIXING OF THE ABOVE VARIANTS AND COMPARING WITH THE NORMAL ANN

w and b site `streamlit for gui`

[119]:

```

```

0.0.4 WEEKLY OUTPUT PDFS

convert notebook to pdf `for weekly progrss submission`

[119]:

```

```

[120]:

```
%cd /content/drive/MyDrive/Colab Notebooks/MTP
!pwd
!ls
```

```
/content/drive/MyDrive/Colab Notebooks/MTP
/content/drive/MyDrive/Colab Notebooks/MTP
 3rd_sem1.pdf                          material
 3rd_sem.pdf                           model_tfp1v1.pkl
 4th_sem_MARCH.pdf                     model_tfp_v1.h5
 4th_sem_mid_FINAL.pdf                 MTP_BNN.ipynb
 4th_sem_mid.pdf                       MTP_BNN.pdf
 4th_sem_mid_plots.pdf                 MTP_Data_Visualization.ipynb
 4th_sem_mid_plots_sir.pdf             READ.md
 4th_sem.pdf                           README.md
'Copy of 4th_sem_mid.pdf'              saved_model
'Copy of 4th_sem_mid_plots.pdf'        w1.pdf
'Copy of 4th_sem_mid_plots_sir.pdf'    w2.pdf
 datasets                             'web app'
 dec.pdf
```

[ ]:

```
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-generic-recommended
```

[122]:

```
!jupyter nbconvert --to pdf --output "4th_sem_mid_FINAL_all" MTP_BNN.ipynb
```

```
[NbConvertApp] Running bibtex 1 time: ['bibtex', './notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 2024552 bytes to 4th_sem_mid_FINAL_all.pdf
```

[122]:

```

```

[122]:

```

```

[122]:

```
# should have saved plots as files for download
```

[123]:

```
from google.colab import files
!zip -r /content/models.zip /content/saved_model
files.download("/content/models.zip")
!zip -r /content/content.zip /content/*.h5
files.download("/content/contenth5.zip")
!zip -r /content/content.zip /content/*.png
```

```
files.download("/content/contentpng.zip")
```

```
 (stored 0%)
  adding: content/drive/MyDrive/Classroom/PM Aug-Dec 2020 Computer Science &
Engineering/Programming Methodology Lab 5: 17th Sep 2020/CS19B047.zip (stored
0%)
  adding: content/drive/MyDrive/Classroom/PM Aug-Dec 2020 Computer Science &
Engineering/Programming Methodology Lab 5: 17th Sep 2020/IMG-20200920-WA0008.jpg
(deflated 0%)
  adding: content/drive/MyDrive/Classroom/PM Aug-Dec 2020 Computer Science &
Engineering/Programming Methodology Lab 5: 17th Sep 2020/CS19B002 AMASA YASWANTH
- Programming Methodology Lab 5: 17th Sep 2020.gslides
zip warning: Operation not supported
        zip warning: could not open for reading:
content/drive/MyDrive/Classroom/PM Aug-Dec 2020 Computer Science &
Engineering/Programming Methodology Lab 5: 17th Sep 2020/CS19B002 AMASA YASWANTH
- Programming Methodology Lab 5: 17th Sep 2020.gslides
  adding: content/drive/MyDrive/Classroom/PM Aug-Dec 2020 Computer Science &
Engineering/Programming Methodology Lab 5: 17th Sep 2020/CS19B002_JAVA2.zip
(stored 0%)
  adding: content/drive/MyDrive/Classroom/PM Aug-Dec 2020 Computer Science &
Engineering/Programming Methodology Lab 5: 17th Sep 2020/CS19B042.txt (deflated
75%)
  adding: content/drive/MyDrive/Classroom/PM Aug-Dec 2020 Computer Science &
Engineering/Programming Methodology Lab 5: 17th Sep 2020/CS19B028_LAB5.zip
(stored 0%)
  adding: content/drive/MyDrive/Classroom/PM Aug-Dec 2020 Computer Science &
Engineering/Programming Methodology Lab 5: 17th Sep 2020/Testcases.txt (deflated
30%)
  adding: content/drive/MyDrive/Classroom/PM Aug-Dec 2020 Computer Science &
Engineering/Programming Methodology Lab 5: 17th Sep 2020/CS19B008 BUKKE ROOPA
SREE - Programming Methodology Lab 5: 17th Sep 2020.gdoc
zip warning: Operation not supported
        zip warning: could not open for reading:
content/drive/MyDrive/Classroom/PM Aug-Dec 2020 Computer Science &
Engineering/Programming Methodology Lab 5: 17th Sep 2020/CS19B008 BUKKE ROOPA
SREE - Programming Methodology Lab 5: 17th Sep 2020.gdoc
  adding: content/drive/MyDrive/Classroom/PM Aug-Dec 2020 Computer Science &
Engineering/Programming Methodology Lab 5: 17th Sep
2020/cs19b008_assignment2_question1.jar (deflated 20%)
  adding: content/drive/MyDrive/Classroom/PM Aug-Dec 2020 Computer Science &
Engineering/Programming Methodology Lab 5: 17th Sep
2020/cs19b008_assignment2_question1 - Shortcut.lnk


zip error: Interrupted (aborting)
```

```
␣
↪---------------------------------------------------------------------------

    FileNotFoundError                         Traceback (most recent call␣
↪last)

    /usr/local/lib/python3.7/dist-packages/pyforest/__init__.py in <module>()
      3 files.download("/content/models.zip")
      4 get_ipython().system('zip -r /content/content.zip /content/')
----> 5 files.download("/content/content.zip")
      6


    /usr/local/lib/python3.7/dist-packages/google/colab/files.py in␣
↪download(filename)
    140       raise OSError(msg)
    141     else:
--> 142       raise FileNotFoundError(msg)  # pylint:␣
↪disable=undefined-variable
    143
    144   comm_manager = _IPython.get_ipython().kernel.comm_manager


    FileNotFoundError: Cannot find file: /content/content.zip
```