In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.datasets
from sklearn.model_selection import train_test_split
```

```
C:\Users\venny\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and
<1.23.0 is required for this version of SciPy (detected version 1.26.0
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
breast_cancer_dataset = sklearn.datasets.load_breast_cancer()
```

In [3]:

```
print(breast_cancer_dataset)
```

```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
         1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]), 'frame': None, 'target_names': array(['mal
ignant', 'benign'], dtype='<U9'), 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnosti
c) dataset\n--------------------------------------------\n\n**Data Set Characteristics:**\n\n    :Number of
Instances: 569\n\n    :Number of Attributes: 30 numeric, predictive attributes and the class\n\n    :Attribu
te Information:\n        - radius (mean of distances from center to points on the perimeter)\n        - text
ure (standard deviation of gray-scale values)\n        - perimeter\n        - area\n        - smoothness (lo
```

```
cal variation in radius lengths)\n        - compactness (perimeter^2 / area - 1.0)\n        - concavity (sev
erity of concave portions of the contour)\n        - concave points (number of concave portions of the conto
ur)\n        - symmetry\n        - fractal dimension ("coastline approximation" - 1)\n\n        The mean, st
andard error, and "worst" or largest (mean of the three\n        worst/largest values) of these features wer
e computed for each image,\n        resulting in 30 features.  For instance, field 0 is Mean Radius, field\n
10 is Radius SE, field 20 is Worst Radius.\n\n        - class:\n                - WDBC-Malignant\n
- WDBC-Benign\n\n    :Summary Statistics:\n\n    ===================================== ====== ======\n
Min    Max\n    ===================================== ====== ======\n    radius (mean):
6.981  28.11\n    texture (mean):                       9.71   39.28\n    perimeter (mean):
43.79  188.5\n    area (mean):                          143.5  2501.0\n    smoothness (mean):
0.053  0.163\n    compactness (mean):                   0.019  0.345\n    concavity (mean):
0.0    0.427\n    concave points (mean):                0.0    0.201\n    symmetry (mean):
0.106  0.304\n    fractal dimension (mean):             0.05   0.097\n    radius (standard error):
0.112  2.873\n    texture (standard error):             0.36   4.885\n    perimeter (standard error):
0.757  21.98\n    area (standard error):                6.802  542.2\n    smoothness (standard error):
0.002  0.031\n    compactness (standard error):         0.002  0.135\n    concavity (standard error):
0.0    0.396\n    concave points (standard error):      0.0    0.053\n    symmetry (standard error):
0.008  0.079\n    fractal dimension (standard error):   0.001  0.03\n    radius (worst):
7.93   36.04\n    texture (worst):                      12.02  49.54\n    perimeter (worst):
50.41  251.2\n    area (worst):                         185.2  4254.0\n    smoothness (worst):
0.071  0.223\n    compactness (worst):                  0.027  1.058\n    concavity (worst):
0.0    1.252\n    concave points (worst):               0.0    0.291\n    symmetry (worst):
0.156  0.664\n    fractal dimension (worst):            0.055  0.208\n    ==================================
=== ====== ======\n\n    :Missing Attribute Values: None\n\n    :Class Distribution: 212 - Malignant, 357 -
Benign\n\n    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian\n\n    :Donor: Nick Str
eet\n\n    :Date: November, 1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) dataset
s.\nhttps://goo.gl/U2Uwz2\n\nFeatures are computed from a digitized image of a fine needle\naspirate (FNA) o
f a breast mass.  They describe\ncharacteristics of the cell nuclei present in the image.\n\nSeparating plan
e described above was obtained using\nMultisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\nConst
ruction Via Linear Programming." Proceedings of the 4th\nMidwest Artificial Intelligence and Cognitive Scien
ce Society,\npp. 97-101, 1992], a classification method which uses linear\nprogramming to construct a decisi
on tree.  Relevant features\nwere selected using an exhaustive search in the space of 1-4\nfeatures and 1-3
separating planes.\n\nThe actual linear program used to obtain the separating plane\nin the 3-dimensional sp
ace is that described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination
of Two Linearly Inseparable Sets",\nOptimization Methods and Software 1, 1992, 23-34].\n\nThis database is a
lso available through the UW CS ftp server:\n\nftp ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/W
DBC/\n\n.. topic:: References\n\n   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extrac
tion \n     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on \n     Electronic Imaging:
Science and Technology, volume 1905, pages 861-870,\n     San Jose, CA, 1993.\n   - O.L. Mangasarian, W.N. S
treet and W.H. Wolberg. Breast cancer diagnosis and \n     prognosis via linear programming. Operations Rese
arch, 43(4), pages 570-577, \n     July-August 1995.\n   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian.
Machine learning techniques\n     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1
994) \n     163-171.', 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
```

```
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23'), 'filename': 'breast_cancer.csv', 'data_m
odule': 'sklearn.datasets.data'}
```

In [4]:
```python
data_frame = pd.DataFrame(breast_cancer_dataset.data,columns = breast_cancer_dataset.feature_names)
```

In [5]:
```python
data_frame.head()
```

Out[5]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	wo perime
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152

5 rows × 30 columns

In [6]:
```python
data_frame['label'] = breast_cancer_dataset.target
```

In [7]:
```
data_frame.tail()
```

Out[7]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	\
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10	2
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00	1
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70	1
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60	1
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.16	

5 rows × 31 columns

In [8]:
```
data_frame.shape
```

Out[8]:
```
(569, 31)
```

In [9]:
```
data_frame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  label                    569 non-null    int32  
dtypes: float64(30), int32(1)
memory usage: 135.7 KB
```

In [10]:
```
data_frame.isnull().sum()
```

Out[10]:
```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
label                      0
dtype: int64
```

In [11]:
```
data_frame.describe()
```

Out[11]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...

8 rows × 31 columns

In [12]:
```
data_frame['label'].value_counts()
```

Out[12]:
```
1    357
0    212
Name: label, dtype: int64
```

In [13]:
```
data_frame.groupby('label').mean()
```

Out[13]:

label	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	wor radi
0	17.462830	21.604906	115.365377	978.376415	0.102898	0.145188	0.160775	0.087990	0.192909	0.062680	...	21.1348
1	12.146524	17.914762	78.075406	462.790196	0.092478	0.080085	0.046058	0.025717	0.174186	0.062867	...	13.3798

2 rows × 30 columns

In [14]:
```
x = data_frame.drop(columns = 'label', axis=1)
y = data_frame['label']
```

In [15]:

```
print(x)
```

```
     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0          17.99         10.38          122.80     1001.0          0.11840
1          20.57         17.77          132.90     1326.0          0.08474
2          19.69         21.25          130.00     1203.0          0.10960
3          11.42         20.38           77.58      386.1          0.14250
4          20.29         14.34          135.10     1297.0          0.10030
..           ...           ...             ...        ...              ...
564        21.56         22.39          142.00     1479.0          0.11100
565        20.13         28.25          131.20     1261.0          0.09780
566        16.60         28.08          108.30      858.1          0.08455
567        20.60         29.33          140.10     1265.0          0.11780
568         7.76         24.54           47.92      181.0          0.05263

     mean compactness  mean concavity  mean concave points  mean symmetry  \
0             0.27760         0.30010              0.14710         0.2419
1             0.07864         0.08690              0.07017         0.1812
2             0.15990         0.19740              0.12790         0.2069
3             0.28390         0.24140              0.10520         0.2597
4             0.13280         0.19800              0.10430         0.1809
..                ...             ...                  ...            ...
564           0.11590         0.24390              0.13890         0.1726
565           0.10340         0.14400              0.09791         0.1752
566           0.10230         0.09251              0.05302         0.1590
567           0.27700         0.35140              0.15200         0.2397
568           0.04362         0.00000              0.00000         0.1587

     mean fractal dimension  ...  worst radius  worst texture  \
0                   0.07871  ...        25.380          17.33
1                   0.05667  ...        24.990          23.41
2                   0.05999  ...        23.570          25.53
3                   0.09744  ...        14.910          26.50
4                   0.05883  ...        22.540          16.67
..                      ...  ...           ...            ...
564                 0.05623  ...        25.450          26.40
565                 0.05533  ...        23.690          38.25
566                 0.05648  ...        18.980          34.12
567                 0.07016  ...        25.740          39.42
568                 0.05884  ...         9.456          30.37

     worst perimeter  worst area  worst smoothness  worst compactness  \
0             184.60      2019.0           0.16220            0.66560
1             158.80      1956.0           0.12380            0.18660
2             152.50      1709.0           0.14440            0.42450
```

```
3          98.87     567.7        0.20980          0.86630
4         152.20    1575.0        0.13740          0.20500
..           ...       ...            ...              ...
564       166.10    2027.0        0.14100          0.21130
565       155.00    1731.0        0.11660          0.19220
566       126.70    1124.0        0.11390          0.30940
567       184.60    1821.0        0.16500          0.86810
568        59.16     268.6        0.08996          0.06444

     worst concavity  worst concave points  worst symmetry  \
0             0.7119                0.2654          0.4601
1             0.2416                0.1860          0.2750
2             0.4504                0.2430          0.3613
3             0.6869                0.2575          0.6638
4             0.4000                0.1625          0.2364
..               ...                   ...             ...
564           0.4107                0.2216          0.2060
565           0.3215                0.1628          0.2572
566           0.3403                0.1418          0.2218
567           0.9387                0.2650          0.4087
568           0.0000                0.0000          0.2871

     worst fractal dimension
0                    0.11890
1                    0.08902
2                    0.08758
3                    0.17300
4                    0.07678
..                       ...
564                  0.07115
565                  0.06637
566                  0.07820
567                  0.12400
568                  0.07039

[569 rows x 30 columns]
```

In [16]:
```
print(y)
```

```
0      0
1      0
2      0
3      0
4      0
      ..
564    0
565    0
566    0
567    0
568    1
Name: label, Length: 569, dtype: int32
```

In [17]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=2)
```

In [18]:
```
print(x.shape, x_train.shape, x_test.shape)
```

```
(569, 30) (455, 30) (114, 30)
```

In [19]:
```
from sklearn.preprocessing import StandardScaler
```

In [20]:
```
Scaler = StandardScaler()

x_train_std = Scaler.fit_transform(x_train)

x_test_std = Scaler.transform(x_test)
```

In [21]:
```
print(x_train_std)
```

```
[[-0.01330339  1.7757658  -0.01491962 ... -0.13236958 -1.08014517
  -0.03527943]
 [-0.8448276  -0.6284278  -0.87702746 ... -1.11552632 -0.85773964
  -0.72098905]
 [ 1.44755936  0.71180168  1.47428816 ...  0.87583964  0.4967602
   0.46321706]
 ...
 [-0.46608541 -1.49375484 -0.53234924 ... -1.32388956 -1.02997851
  -0.75145272]
 [-0.50025764 -1.62161319 -0.527814   ... -0.0987626   0.35796577
  -0.43906159]
 [ 0.96060511  1.21181916  1.00427242 ...  0.8956983  -1.23064515
   0.50697397]]
```

In [22]:
```
import tensorflow as tf
tf.random.set_seed(3)
from tensorflow import keras
```

In [23]:
```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(30,)),
    keras.layers.Dense(20,activation='relu'),
    keras.layers.Dense(2,activation='sigmoid'),
])
```

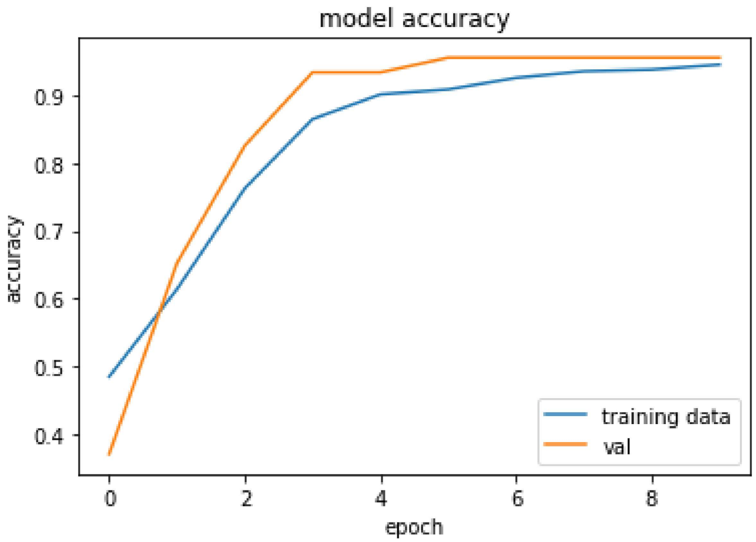
In [24]:
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [25]:
```python
history = model.fit(x_train_std, y_train, validation_split=0.1, epochs=10)
```

```
Epoch 1/10
13/13 [==============================] - 1s 19ms/step - loss: 0.9900 - accuracy: 0.4841 - val_loss: 0.9318 -
val_accuracy: 0.3696
Epoch 2/10
13/13 [==============================] - 0s 3ms/step - loss: 0.6383 - accuracy: 0.6137 - val_loss: 0.5957 -
val_accuracy: 0.6522
Epoch 3/10
13/13 [==============================] - 0s 4ms/step - loss: 0.4349 - accuracy: 0.7628 - val_loss: 0.4152 -
val_accuracy: 0.8261
Epoch 4/10
13/13 [==============================] - 0s 3ms/step - loss: 0.3233 - accuracy: 0.8655 - val_loss: 0.3125 -
val_accuracy: 0.9348
Epoch 5/10
13/13 [==============================] - 0s 3ms/step - loss: 0.2581 - accuracy: 0.9022 - val_loss: 0.2547 -
val_accuracy: 0.9348
Epoch 6/10
13/13 [==============================] - 0s 4ms/step - loss: 0.2188 - accuracy: 0.9095 - val_loss: 0.2175 -
val_accuracy: 0.9565
Epoch 7/10
13/13 [==============================] - 0s 4ms/step - loss: 0.1920 - accuracy: 0.9267 - val_loss: 0.1936 -
val_accuracy: 0.9565
Epoch 8/10
13/13 [==============================] - 0s 4ms/step - loss: 0.1727 - accuracy: 0.9364 - val_loss: 0.1790 -
val_accuracy: 0.9565
Epoch 9/10
13/13 [==============================] - 0s 4ms/step - loss: 0.1586 - accuracy: 0.9389 - val_loss: 0.1671 -
val_accuracy: 0.9565
Epoch 10/10
13/13 [==============================] - 0s 2ms/step - loss: 0.1471 - accuracy: 0.9462 - val_loss: 0.1584 -
val_accuracy: 0.9565
```

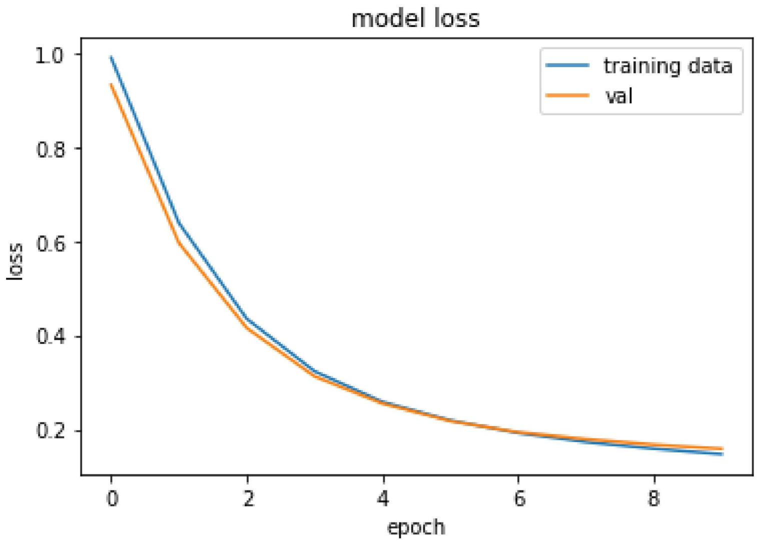
In [26]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['training data', 'val'], loc = 'lower right')
```

Out[26]:
```
<matplotlib.legend.Legend at 0x2c193b750d0>
```



In [27]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['training data', 'val'], loc = 'upper right')
```

Out[27]: <matplotlib.legend.Legend at 0x2c193b34670>



In [28]:
```
loss, accuracy = model.evaluate(x_test_std, y_test)
print(accuracy)
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.1467 - accuracy: 0.9474
0.9473684430122375
```

In [29]:
```
print(x_test_std.shape)
print(x_test_std[0])
```

```
(114, 30)
[-0.04462793 -1.41612656 -0.05903514 -0.16234067  2.0202457  -0.11323672
  0.18500609  0.47102419  0.63336386  0.26335737  0.53209124  2.62763999
  0.62351167  0.11405261  1.01246781  0.41126289  0.63848593  2.88971815
 -0.41675911  0.74270853 -0.32983699 -1.67435595 -0.36854552 -0.38767294
  0.32655007 -0.74858917 -0.54689089 -0.18278004 -1.23064515 -0.6268286 ]
```

In [30]:
```
y_pred = model.predict(x_test_std)
```

```
4/4 [==============================] - 0s 3ms/step
```

In [31]:
```
print(y_pred.shape)
print(y_pred[0])
```

```
(114, 2)
[0.08001052 0.15083475]
```

In [32]:
```
print(x_test_std)
```

```
[[-0.04462793 -1.41612656 -0.05903514 ... -0.18278004 -1.23064515
  -0.6268286 ]
 [ 0.24583601 -0.06219797  0.21802678 ...  0.54129749  0.11047691
   0.0483572 ]
 [-1.26115925 -0.29051645 -1.26499659 ... -1.35138617  0.269338
  -0.28231213]
 ...
 [ 0.72709489  0.45836817  0.75277276 ...  1.46701686  1.19909344
   0.65319961]
 [ 0.25437907  1.33054477  0.15659489 ... -1.29043534 -2.22561725
  -1.59557344]
 [ 0.84100232 -0.06676434  0.8929529  ...  2.15137705  0.35629355
   0.37459546]]
```

In [33]:
```
print(y_pred)
```

```
[[0.08001052 0.15083475]
 [0.48364252 0.5798445 ]
 [0.05290874 0.8882098 ]
 [0.9966045  0.00143801]
 [0.5151394  0.5583441 ]
 [0.980065   0.01517438]
 [0.23418538 0.52150357]
 [0.09316273 0.8624504 ]
 [0.11148371 0.7898397 ]
 [0.17932348 0.6722547 ]
 [0.41614878 0.6115913 ]
 [0.08939689 0.84544945]
 [0.07990435 0.42755887]
 [0.16911446 0.7787212 ]
 [0.14161026 0.806029  ]
 [0.8372073  0.04663235]
 [0.09838139 0.843916  ]
 [0.02013074 0.4916801 ]
 [0.12646021 0.56251097]
```

In [34]:
```
my_list = [0.25, 0.56]

index_of_max_value = np.argmax(my_list)
print(my_list)
print(index_of_max_value)
```

```
[0.25, 0.56]
1
```

In [35]:
```
y_pred_labels = [np.argmax(i) for i in y_pred]
print(y_pred_labels)
```

```
[1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,
1, 1, 0, 0, 0, 0]
```

In [ ]: