Notebook _ranking.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_auc_score
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
```

```
df = pd.read_csv("glass.csv")
df.head()
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

```
df.isnull().sum()
```

```
RI      0
Na      0
Mg      0
Al      0
Si      0
K       0
Ca      0
Ba      0
Fe      0
Type    0
dtype: int64
```

```
df.shape
```

```
(214, 10)
```

```
x= df.iloc[:,:-1].values
y=df.iloc[:,9].values
```

```
x
```

```
array([[ 1.52101, 13.64   ,  4.49   , ...,  8.75   ,  0.     ,  0.     ],
```

```
       [ 1.51761, 13.89   ,  3.6    , ...,  7.83   ,  0.     ,  0.     ],
       [ 1.51618, 13.53   ,  3.55   , ...,  7.78   ,  0.     ,  0.     ],
       ...,
       [ 1.52065, 14.36   ,  0.     , ...,  8.44   ,  1.64   ,  0.     ],
       [ 1.51651, 14.38   ,  0.     , ...,  8.48   ,  1.57   ,  0.     ],
       [ 1.51711, 14.23   ,  0.     , ...,  8.62   ,  1.67   ,  0.     ]])
```

```
y
```

```
array([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, 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
       7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7])
```

```
df.head()
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

```
ss = StandardScaler()
x = ss.fit_transform(x)
```

```
x_train ,x_test, y_train,y_test = train_test_split(x,y,test_size=0.3)
```

```
x.shape
```

```
(214, 9)
```

```
x
```

```
array([[ 0.87286765,  0.28495326,  1.25463857, ..., -0.14576634,
        -0.35287683, -0.5864509 ],
       [-0.24933347,  0.59181718,  0.63616803, ..., -0.79373376,
        -0.35287683, -0.5864509 ],
       [-0.72131806,  0.14993314,  0.60142249, ..., -0.82894938,
        -0.35287683, -0.5864509 ],
       ...,
```

```
            [ 0.75404635,  1.16872135, -1.86551055, ..., -0.36410319,
              2.95320036, -0.5864509 ],
            [-0.61239854,  1.19327046, -1.86551055, ..., -0.33593069,
              2.81208731, -0.5864509 ],
            [-0.41436305,  1.00915211, -1.86551055, ..., -0.23732695,
              3.01367739, -0.5864509 ]])
```

```
y
```

```
array([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, 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
       7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7])
```

```
x_train
```

```
array([[ 0.16984165,  1.94201842, -1.32348019, ...,  0.69940856,
        -0.35287683, -0.5864509 ],
       [ 0.32166886,  0.28495326,  0.59447339, ..., -0.0471626 ,
        -0.05049172,  1.88241125],
       [-0.14371454, -0.24285268,  0.55277874, ..., -0.37114631,
        -0.35287683, -0.5864509 ],
       ...,
       [ 0.87286765,  0.28495326,  1.25463857, ..., -0.14576634,
        -0.35287683, -0.5864509 ],
       [-0.79393107, -0.75838406,  0.64311714, ..., -0.62469878,
        -0.35287683,  2.08814977],
       [ 1.10720964,  0.08856035,  0.73345553, ...,  0.45994234,
        -0.35287683,  0.13363389]])
```

```
y_train
```

```
array([6, 3, 1, 3, 1, 1, 2, 2, 1, 7, 2, 2, 1, 2, 3, 5, 2, 1, 2, 7, 2, 7,
       1, 6, 3, 2, 7, 2, 2, 1, 1, 1, 7, 2, 5, 2, 2, 5, 1, 3, 7, 2, 1, 5,
       7, 2, 2, 1, 2, 2, 1, 5, 3, 7, 1, 2, 1, 1, 2, 1, 2, 1, 1, 3, 1, 1,
       2, 5, 2, 2, 1, 5, 1, 2, 7, 1, 2, 2, 7, 1, 2, 6, 2, 2, 2, 6, 2, 1,
       3, 7, 6, 1, 1, 2, 3, 6, 2, 1, 2, 2, 2, 7, 2, 2, 7, 1, 1, 2, 1, 5,
       2, 2, 5, 2, 5, 3, 1, 3, 2, 2, 1, 2, 2, 1, 2, 3, 1, 7, 2, 3, 1, 1,
       1, 2, 2, 2, 2, 1, 7, 1, 7, 2, 7, 1, 2, 1, 1, 1, 1])
```

```
x_test
```

```
             3.20234454e+00, -2.80942033e+00,  8.78014487e+00,
            -1.40648556e+00, -3.52876828e-01, -5.86450902e-01],
           [-6.09097944e-01,  1.86756807e-01,  6.15320709e-01,
             5.03781754e-02, -2.60030725e-01,  2.19688551e-01,
            -7.02173148e-01, -3.52876828e-01, -5.86450902e-01],
           [ 1.74422381e+00,  2.92398296e+00, -5.93823938e-01,
            -2.70841327e-01, -2.87412565e+00, -2.87483906e-01,
            -2.44370078e-01,  3.03383639e+00, -5.86450902e-01],
           [ 3.31570637e-01,  4.69071613e-01, -1.90775722e-01,
            -5.11755954e-01,  1.41142259e-01, -7.63918639e-01,
```

```
  5.72632323e-01, -3.52876828e-01, -5.86450902e-01],
[-5.66190255e-01, -3.28774576e-01,  5.31931423e-01,
  6.32588523e-01, -2.21207533e-01,  1.58213102e-01,
 -4.06361934e-01, -3.52876828e-01,  1.16232646e+00],
[-3.35148849e-01, -4.76069257e-01,  5.94473387e-01,
  4.92054991e-01,  1.02319067e-01,  6.59999275e-02,
 -3.64103189e-01, -3.52876828e-01,  1.33633894e-01],
[ 2.74100244e+00,  7.14562748e-01,  7.05659102e-01,
 -1.47541446e+00, -1.39884436e+00, -7.33180914e-01,
  6.07847943e-01, -3.52876828e-01,  4.42241664e-01],
[-5.45985699e-02, -9.67051528e-01,  5.24430284e-02,
 -1.23449983e+00,  1.49995398e+00, -2.26008457e-01,
  3.26122977e-01, -3.52876828e-01,  1.47093423e+00],
[-6.05797353e-01, -1.78944683e+00, -1.86551055e+00,
 -5.11755954e-01,  3.27287974e+00,  3.38567419e+00,
 -1.89901058e-02, -3.52876828e-01, -5.86450902e-01],
[ 2.25951709e-01,  2.35855034e-01,  6.50066245e-01,
 -6.72365705e-01, -1.41505092e-02, -5.48754566e-01,
 -1.38723216e-01, -3.52876828e-01, -5.86450902e-01],
[-2.05145644e+00, -5.12892928e-01,  5.45829637e-01,
 -6.52289486e-01,  4.25845666e-01,  1.88950826e-01,
 -4.27491306e-01, -3.52876828e-01,  2.60249605e+00],
[-5.62889663e-01, -6.72462165e-01, -1.86551055e+00,
  7.73122056e-01,  1.59054143e+00,  7.26861008e-01,
  8.54357288e-01, -3.52876828e-01, -5.86450902e-01],
[-7.24618648e-01,  1.89292020e+00, -1.86551055e+00,
  1.65647569e+00,  8.39959714e-01, -7.63918639e-01,
 -1.73938837e-01,  9.97776659e-01, -5.86450902e-01],
[-1.49914717e-02, -9.55579979e-02,  4.55491244e-01,
  1.90911708e-01, -6.61203709e-01,  9.67376522e-02,
  2.32686391e-02, -3.52876828e-01, -5.86450902e-01],
[-6.11997530e-02,  3.83149716e-01, -1.86551055e+00,
 -1.77655774e+00,  2.32818207e+00, -7.63918639e-01,
  1.43189347e+00, -3.52876828e-01, -5.86450902e-01],
[ 3.45174010e-02, -3.77872803e-01,  8.93284995e-01,
 -5.11755954e-01, -2.72971789e-01,  1.58213102e-01,
 -3.71146313e-01, -3.52876828e-01, -5.86450902e-01],
[ 1.52968536e+00, -1.20107111e-01,  6.22269816e-01,
 -1.25457605e+00, -8.55319668e-01, -5.79492291e-01,
  8.54357288e-01, -3.52876828e-01, -2.77843132e-01],
[-1.66818681e-01, -8.32031403e-01,  5.18033208e-01,
 -5.11755954e-01,  3.87022474e-01,  1.88950826e-01,
 -1.38723216e-01, -3.52876828e-01,  2.49962679e+00],
[-5.59589072e-01, -5.74265711e-01,  6.43117137e-01,
 -3.71222421e-01,  1.28201195e-01,  9.67376522e-02,
 -2.51413203e-01, -3.52876828e-01, -5.86450902e-01],

[-4.20964228e-01, -5.00618371e-01,  5.45829637e-01,
  6.92817180e-01,  8.93780027e-02,  2.50426276e-01,
 -5.47224417e-01, -3.52876828e-01, -5.86450902e-01]])
```

```
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score

clf = KNeighborsClassifier(n_neighbors=5)

clf.fit(x_train,y_train)

preds = clf.predict(x_test)
```

```
accuracy_score(y_test,y_hat)
```

```
0.35384615384615387
```

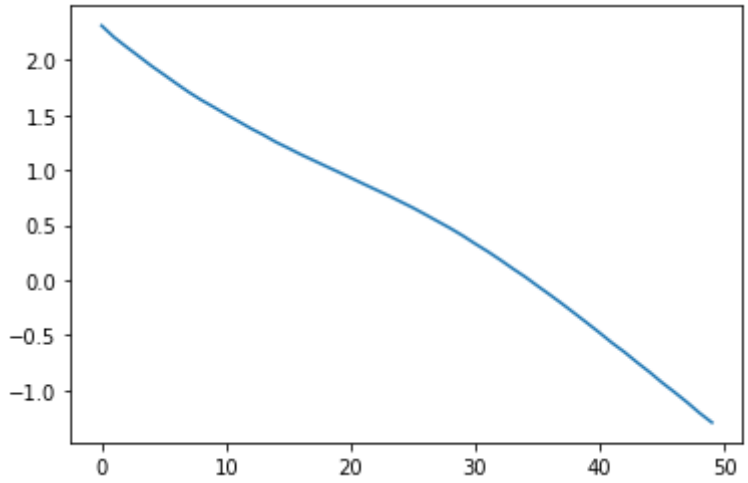
```
model = tf.keras.Sequential([
                             tf.keras.layers.Dense(3,activation= "tanh", input_shap
                             tf.keras.layers.Dense(2,activation= "relu", kernel_reg
                             tf.keras.layers.Dense(1,activation="sigmoid")
                             ])
```

```
model.compile(optimizer="adam", loss="binary_crossentropy")
trained_model = model.fit(x_train, y_train, epochs=50, batch_size=20)
```

```
8/8 [==============================] - 0s 2ms/step - loss: 0.9303
Epoch 22/50
8/8 [==============================] - 0s 2ms/step - loss: 0.8765
Epoch 23/50
8/8 [==============================] - 0s 3ms/step - loss: 0.8235
Epoch 24/50
8/8 [==============================] - 0s 2ms/step - loss: 0.7703
Epoch 25/50
8/8 [==============================] - 0s 3ms/step - loss: 0.7142
Epoch 26/50
8/8 [==============================] - 0s 2ms/step - loss: 0.6581
Epoch 27/50
8/8 [==============================] - 0s 2ms/step - loss: 0.5967
Epoch 28/50
8/8 [==============================] - 0s 3ms/step - loss: 0.5344
Epoch 29/50
8/8 [==============================] - 0s 3ms/step - loss: 0.4722
Epoch 30/50
8/8 [==============================] - 0s 3ms/step - loss: 0.4047
Epoch 31/50
8/8 [==============================] - 0s 2ms/step - loss: 0.3313
Epoch 32/50
8/8 [==============================] - 0s 4ms/step - loss: 0.2616
Epoch 33/50
8/8 [==============================] - 0s 2ms/step - loss: 0.1863
Epoch 34/50
8/8 [==============================] - 0s 2ms/step - loss: 0.1072
Epoch 35/50
8/8 [==============================] - 0s 2ms/step - loss: 0.0317
Epoch 36/50
8/8 [==============================] - 0s 2ms/step - loss: -0.0512
Epoch 37/50
8/8 [==============================] - 0s 2ms/step - loss: -0.1321
Epoch 38/50
8/8 [==============================] - 0s 3ms/step - loss: -0.2154
Epoch 39/50
8/8 [==============================] - 0s 2ms/step - loss: -0.3022
Epoch 40/50
8/8 [==============================] - 0s 2ms/step - loss: -0.3892
Epoch 41/50
8/8 [==============================] - 0s 3ms/step - loss: -0.4780
Epoch 42/50
8/8 [==============================] - 0s 3ms/step - loss: -0.5698
Epoch 43/50
```

```
Epoch 43/50
8/8 [==============================] - 0s 2ms/step - loss: -0.6538
Epoch 44/50
8/8 [==============================] - 0s 2ms/step - loss: -0.7461
Epoch 45/50
8/8 [==============================] - 0s 3ms/step - loss: -0.8341
Epoch 46/50
8/8 [==============================] - 0s 2ms/step - loss: -0.9292
Epoch 47/50
8/8 [==============================] - 0s 2ms/step - loss: -1.0164
Epoch 48/50
8/8 [==============================] - 0s 2ms/step - loss: -1.1070
Epoch 49/50
8/8 [==============================] - 0s 2ms/step - loss: -1.2043
Epoch 50/50
8/8 [==============================] - 0s 2ms/step - loss: -1.2917
```

```
plt.plot(trained_model.history['loss'])
plt.show()
```




```
y_hat = model.predict(x_test)
```

```
y_hat
```

```
[0.79821885],
[0.84662956],
[0.84203833],
[0.8827684 ],
[0.85014653],
[0.8832141 ],
[0.750188  ],
[0.5938172 ],
[0.89201987],
[0.8425747 ],
[0.86525846],
[0.5938172 ],
[0.5840199 ],
[0.88496387],
[0.8424794 ],
[0.61704606],
[0.8623701 ],
[0.88253045],
```

```
              [0.8698125 ],
              [0.72409415],
              [0.87711036],
              [0.5654908 ],
              [0.8224378 ],
              [0.57255036],
              [0.8828591 ],
              [0.86282134],
              [0.878672  ],
              [0.5938172 ],
              [0.8814895 ],
              [0.8588251 ],
              [0.7903329 ],
              [0.8657908 ],
              [0.72074884],
              [0.55685943],
              [0.862505  ],
              [0.5938172 ],
              [0.69158036],
              [0.8410388 ],
              [0.827345  ],
              [0.5938172 ],
              [0.8482418 ],
              [0.570366  ],
              [0.7707666 ],
              [0.6180214 ],
              [0.79536927],
              [0.5570481 ],
              [0.830189  ],
              [0.8909489 ],
              [0.7961799 ],
              [0.7244579 ],
              [0.88370854],
              [0.88137287],
              [0.70294344],
              [0.8188897 ],
              [0.7921199 ],
              [0.5619817 ],
              [0.5938172 ],
              [0.8553549 ],
              [0.85275126]], dtype=float32)
```

```
y_hat = np.where(y_hat>=0.5, 1, 0).flatten()
```

```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

```
classifier = Sequential()
```

```
model = tf.keras.Sequential([
                             tf.keras.layers.Dense(3,activation= "sigmoid", input_s
                             tf.keras.layers.Dense(3,activation= "relu"),
                             tf.keras.layers.Dropout(0.2),
                             tf.keras.layers.Dense(1,activation="tanh")
```

```
])
```

```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_21 (Dense)             (None, 3)                 30        
_________________________________________________________________
dense_22 (Dense)             (None, 3)                 12        
_________________________________________________________________
dropout_1 (Dropout)          (None, 3)                 0         
_________________________________________________________________
dense_23 (Dense)             (None, 1)                 4         
=================================================================
Total params: 46
Trainable params: 46
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer='adam', loss="binary_crossentropy")
trained_model = model.fit(x_train, y_train, epochs=100,batch_size=10)
```

```
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 72/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 73/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 74/100
15/15 [==============================] - 0s 3ms/step - loss: 41.7198
Epoch 75/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 76/100
15/15 [==============================] - 0s 1ms/step - loss: 41.7198
Epoch 77/100
15/15 [==============================] - 0s 1ms/step - loss: 41.7198
Epoch 78/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 79/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 80/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 81/100
15/15 [==============================] - 0s 1ms/step - loss: 41.7198
Epoch 82/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 83/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 84/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 85/100
15/15 [==============================] - 0s 1ms/step - loss: 41.7198
Epoch 86/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 87/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 88/100
```

```
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 89/100
15/15 [==============================] - 0s 1ms/step - loss: 41.7198
Epoch 90/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 91/100
15/15 [==============================] - 0s 1ms/step - loss: 41.7198
Epoch 92/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 93/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 94/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 95/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 96/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 97/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 98/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 99/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
Epoch 100/100
15/15 [==============================] - 0s 2ms/step - loss: 41.7198
```

```
y_hat1 = model.predict(x_test)
y_hat1
```

```
            [-0.1250750],
            [-0.2559656 ],
            [-0.2732325 ],
            [-0.23509595],
            [-0.3299277 ],
            [-0.27193892],
            [-0.22497869],
            [-0.1468645 ],
            [-0.25379705],
            [-0.21946089],
            [-0.21292333],
            [-0.19671497],
            [-0.25744772],
            [-0.31515133],
            [-0.24147807],
            [-0.21526086],
            [-0.21494502],
            [-0.2260948 ],
            [-0.36016193],
            [-0.31661433],
            [-0.21872628],
            [-0.24201918],
            [-0.20000193],
            [-0.2681568 ],
            [-0.36076406],
            [-0.27135193],
            [-0.20336787],
            [-0.26018032],
            [-0.21936336],
            [-0.2379795 ],
            [-0.22845986],
            [-0.16764209],
```

```
       [-0.267579  ],
       [-0.1625808 ],
       [-0.34805614],
       [-0.20920493],
       [-0.24278139],
       [-0.26794782],
       [-0.32303822],
       [-0.24403825],
       [-0.11459293],
       [-0.25077504],
       [-0.26747304],
       [-0.28719655],
       [-0.22727968],
       [-0.2181662 ],
       [-0.32755032],
       [-0.2188855 ],
       [-0.17992193],
       [-0.38631916],
       [-0.29552174],
       [-0.26939055],
       [-0.24022906],
       [-0.22951104],
       [-0.23237355],
       [-0.21699664],
       [-0.23968904],
       [-0.31014436]], dtype=float32)
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_hat))
```

```
              precision    recall  f1-score   support

           1       0.35      1.00      0.52        23
           2       0.00      0.00      0.00        20
           3       0.00      0.00      0.00         4
           5       0.00      0.00      0.00         3
           6       0.00      0.00      0.00         3
           7       0.00      0.00      0.00        12

    accuracy                           0.35        65
   macro avg       0.06      0.17      0.09        65
weighted avg       0.13      0.35      0.18        65

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_hat)
```

```
0.35384615384615387
```

✓ 0s completed at 6:50 PM