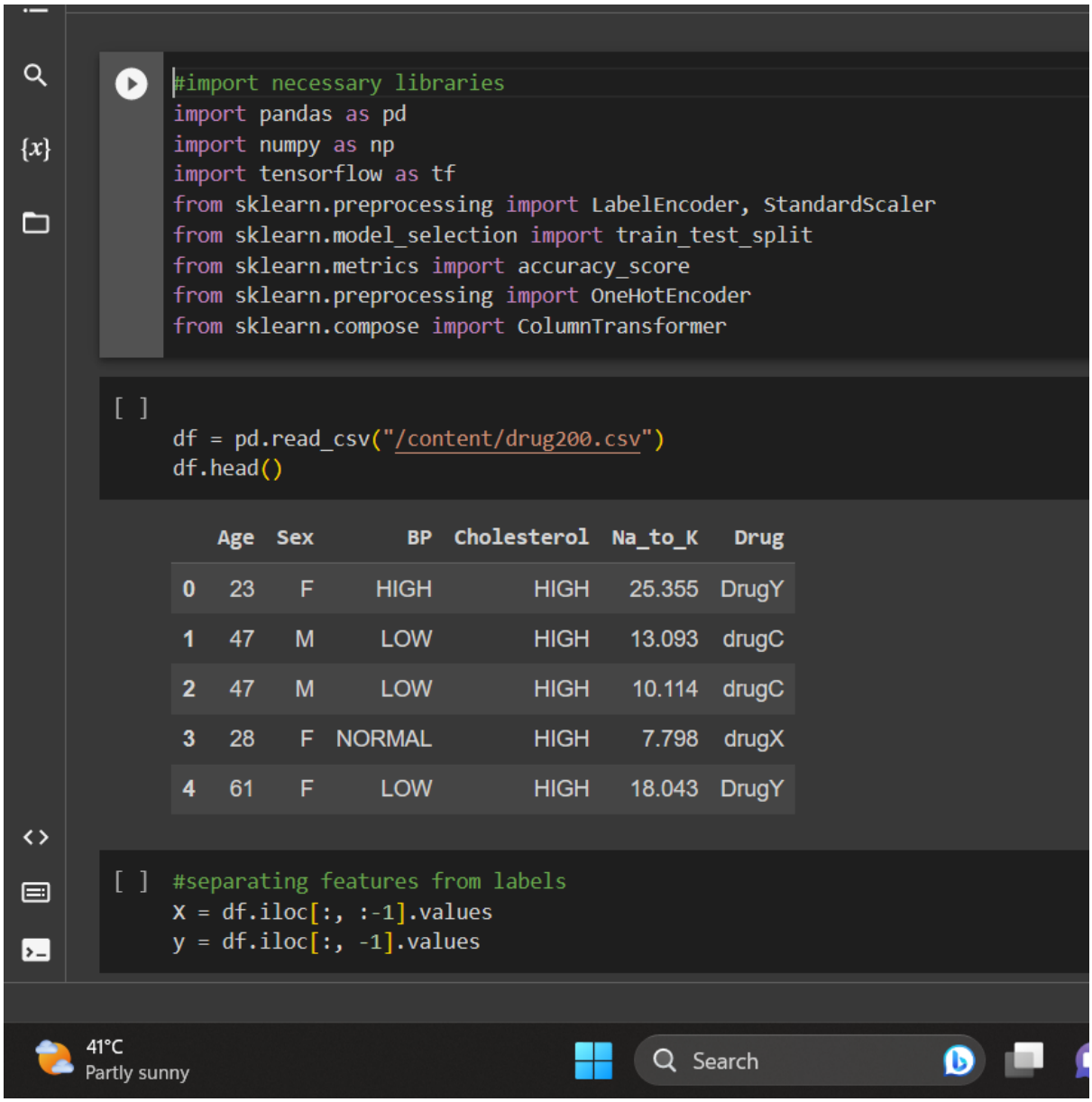
Assignment 2

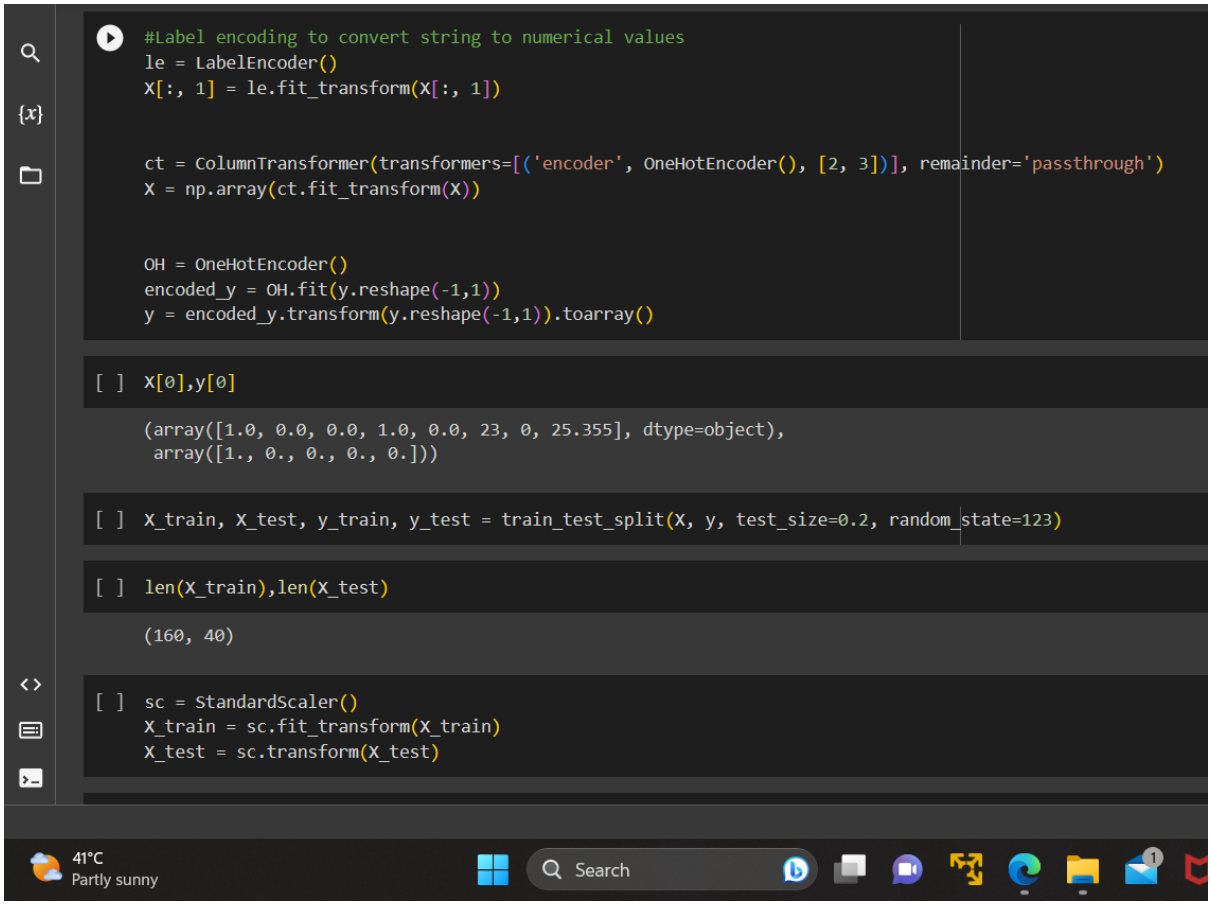


```
#import necessary libraries
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

```
[ ]
df = pd.read_csv("/content/drug200.csv")
df.head()
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	DrugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	DrugY

```
[ ] #separating features from labels
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

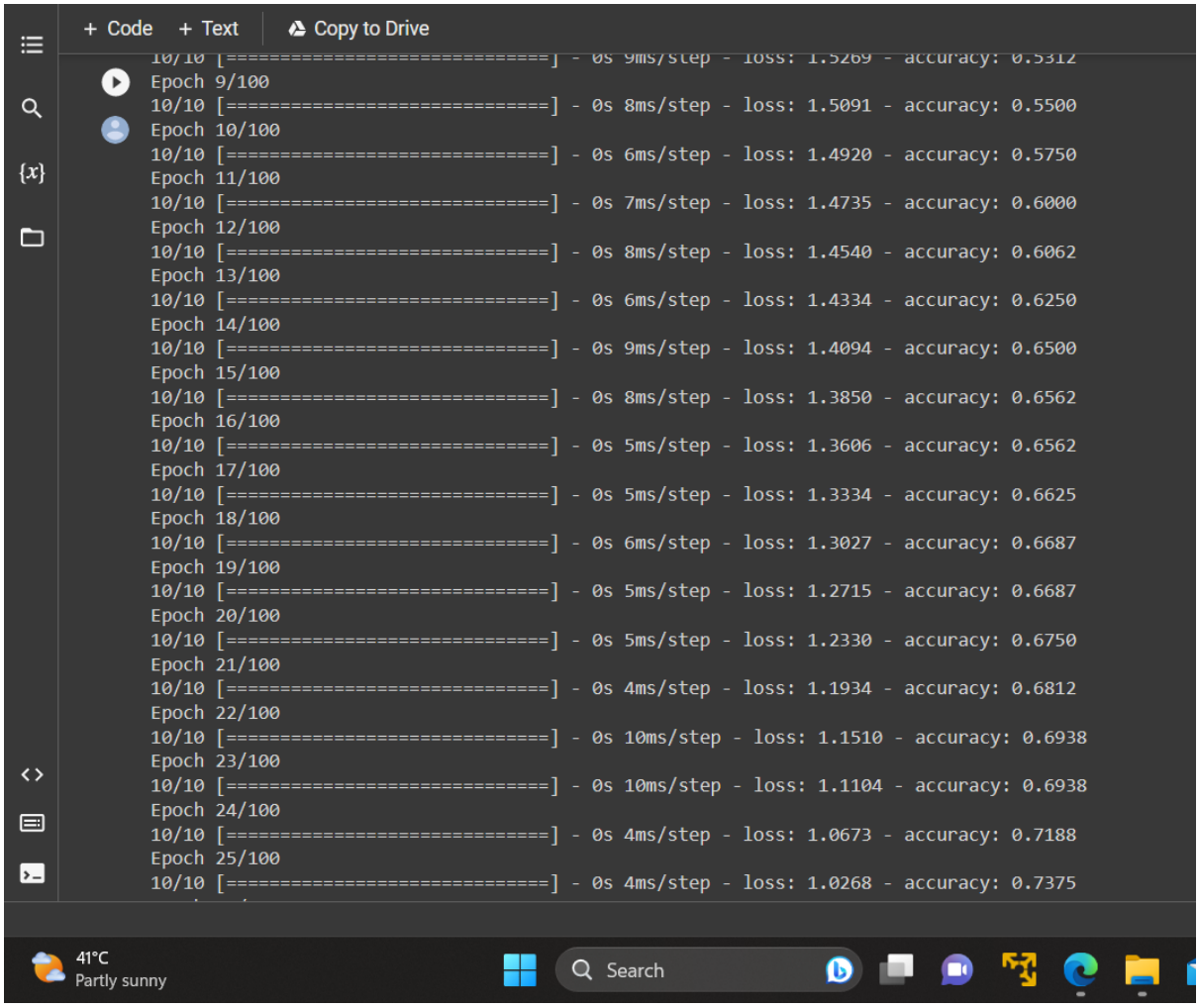

#Label encoding to convert string to numerical values
le = LabelEncoder()
X[:, 1] = le.fit_transform(X[:, 1])
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [2, 3])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
OH = OneHotEncoder()
encoded_y = OH.fit(y.reshape(-1,1))
y = encoded_y.transform(y.reshape(-1,1)).toarray()
[ ] X[0],y[0]
(array([1.0, 0.0, 0.0, 1.0, 0.0, 23, 0, 25.355], dtype=object),
array([1., 0., 0., 0., 0.]))
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
[ ] len(X_train),len(X_test)
(160, 40)
[ ] sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
41°C
Partly sunny
Search

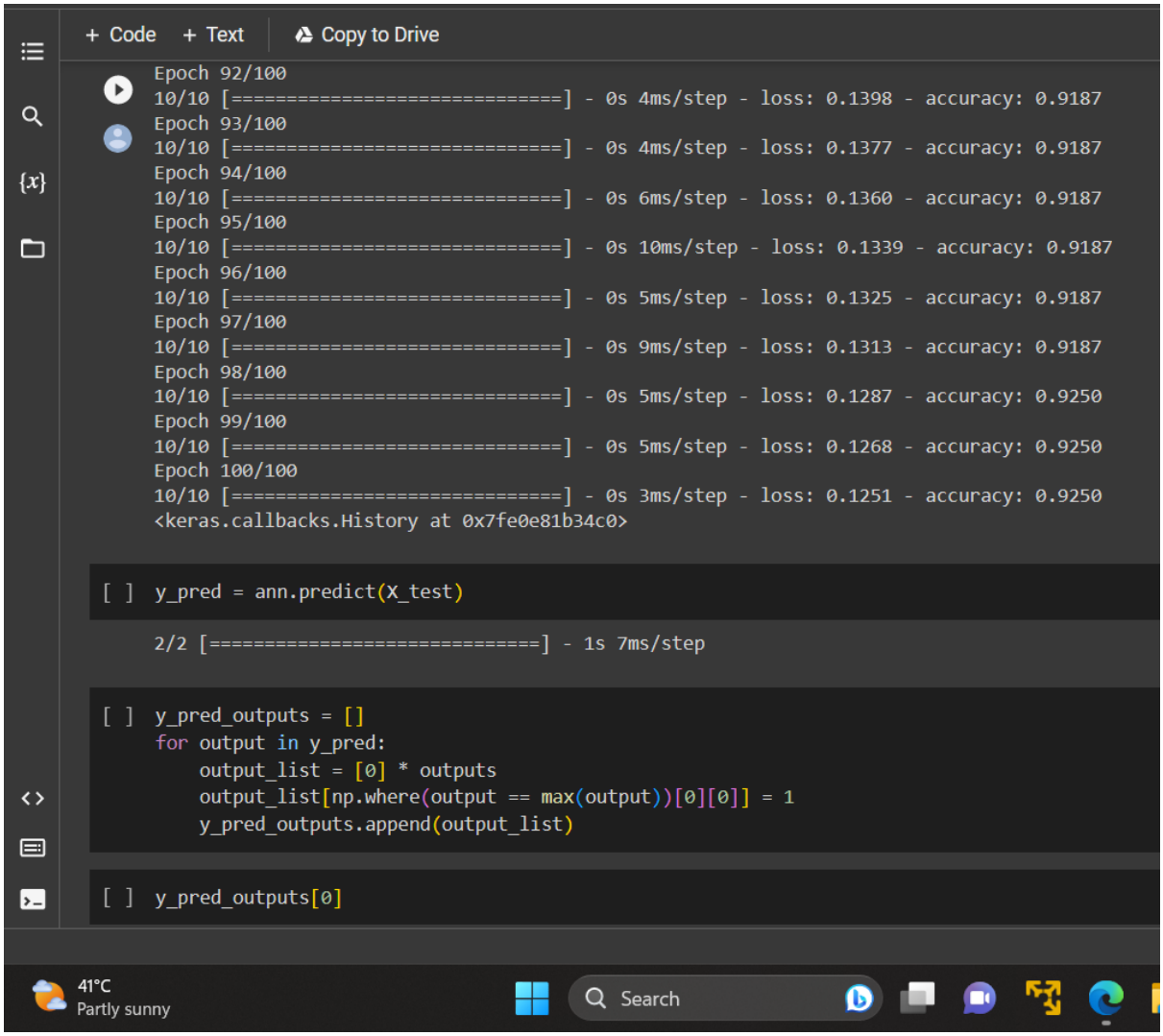
```
ann = tf.keras.models.Sequential()
ann.add(tf.keras.layers.Dense(units=15, activation='relu'))
ann.add(tf.keras.layers.Dense(units=10, activation='relu'))
ann.add(tf.keras.layers.Dense(units=5, activation='relu'))
outputs =  len(set(df["Drug"]))
ann.add(tf.keras.layers.Dense(units=outputs, activation='softmax'))
```

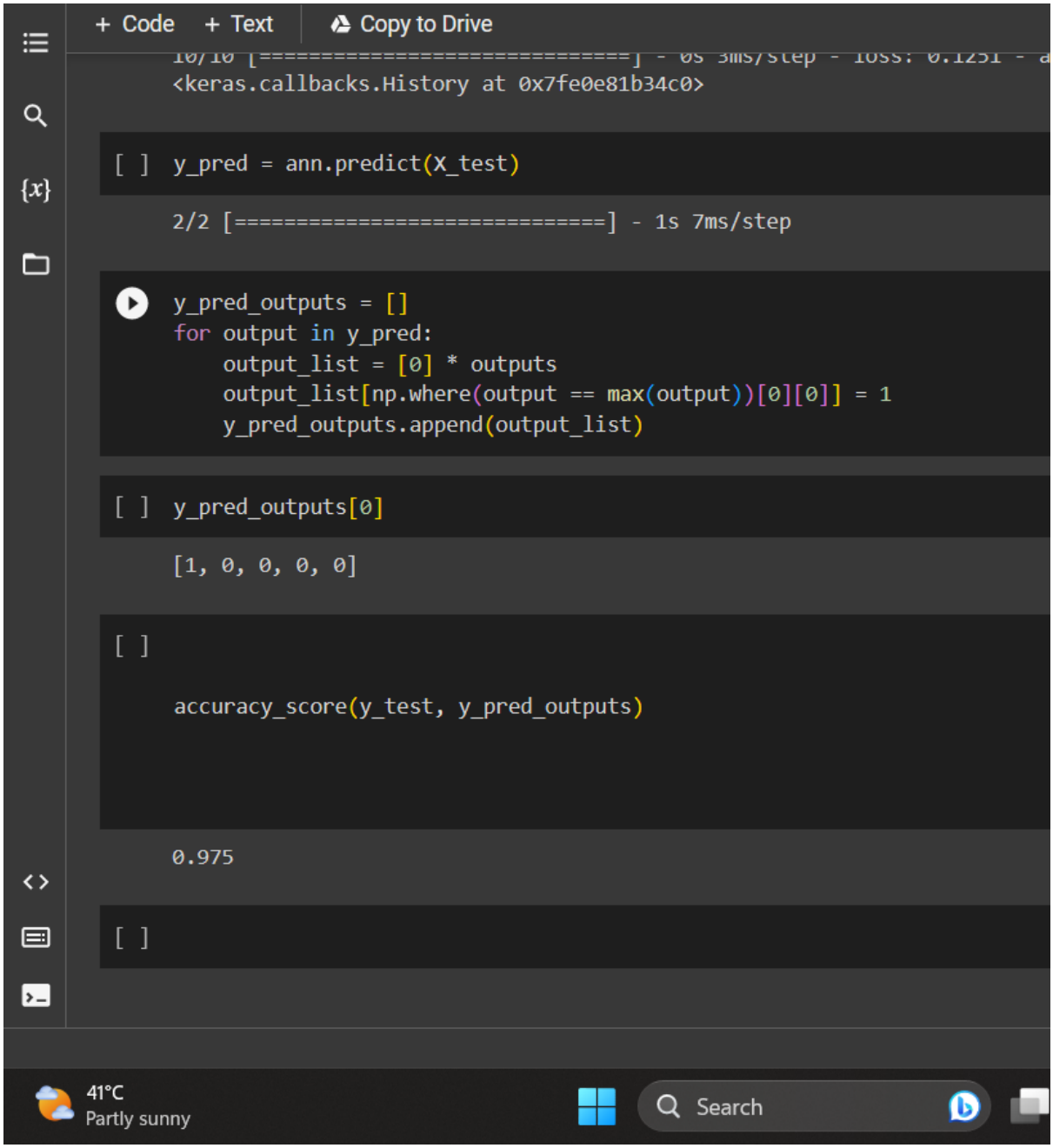
```
[ ] ann.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
[ ] ann.fit(X_train, y_train, batch_size=16, epochs=100)
```

```
Epoch 1/100
10/10 [==============================] - 2s 12ms/step - loss: 1.7858 - accuracy: 0.1063
Epoch 2/100
10/10 [==============================] - 0s 7ms/step - loss: 1.7048 - accuracy: 0.1375
Epoch 3/100
10/10 [==============================] - 0s 9ms/step - loss: 1.6533 - accuracy: 0.1937
Epoch 4/100
10/10 [==============================] - 0s 5ms/step - loss: 1.6164 - accuracy: 0.2750
Epoch 5/100
10/10 [==============================] - 0s 8ms/step - loss: 1.5874 - accuracy: 0.3313
Epoch 6/100
10/10 [==============================] - 0s 7ms/step - loss: 1.5648 - accuracy: 0.4062
Epoch 7/100
10/10 [==============================] - 0s 7ms/step - loss: 1.5457 - accuracy: 0.4750
Epoch 8/100
10/10 [==============================] - 0s 9ms/step - loss: 1.5269 - accuracy: 0.5312
Epoch 9/100
10/10 [==============================] - 0s 8ms/step - loss: 1.5091 - accuracy: 0.5500
Epoch 10/100
10/10 [==============================] - 0s 6ms/step - loss: 1.4920 - accuracy: 0.5750
Epoch 11/100
```


+ Code
+ Text
Copy to Drive
10/10 [==========] - 0s 9ms/step - loss: 1.5269 - accuracy: 0.5312
Epoch 9/100
10/10 [==========] - 0s 8ms/step - loss: 1.5091 - accuracy: 0.5500
Epoch 10/100
10/10 [==========] - 0s 6ms/step - loss: 1.4920 - accuracy: 0.5750
Epoch 11/100
10/10 [==========] - 0s 7ms/step - loss: 1.4735 - accuracy: 0.6000
Epoch 12/100
10/10 [==========] - 0s 8ms/step - loss: 1.4540 - accuracy: 0.6062
Epoch 13/100
10/10 [==========] - 0s 6ms/step - loss: 1.4334 - accuracy: 0.6250
Epoch 14/100
10/10 [==========] - 0s 9ms/step - loss: 1.4094 - accuracy: 0.6500
Epoch 15/100
10/10 [==========] - 0s 8ms/step - loss: 1.3850 - accuracy: 0.6562
Epoch 16/100
10/10 [==========] - 0s 5ms/step - loss: 1.3606 - accuracy: 0.6562
Epoch 17/100
10/10 [==========] - 0s 5ms/step - loss: 1.3334 - accuracy: 0.6625
Epoch 18/100
10/10 [==========] - 0s 6ms/step - loss: 1.3027 - accuracy: 0.6687
Epoch 19/100
10/10 [==========] - 0s 5ms/step - loss: 1.2715 - accuracy: 0.6687
Epoch 20/100
10/10 [==========] - 0s 5ms/step - loss: 1.2330 - accuracy: 0.6750
Epoch 21/100
10/10 [==========] - 0s 4ms/step - loss: 1.1934 - accuracy: 0.6812
Epoch 22/100
10/10 [==========] - 0s 10ms/step - loss: 1.1510 - accuracy: 0.6938
Epoch 23/100
10/10 [==========] - 0s 10ms/step - loss: 1.1104 - accuracy: 0.6938
Epoch 24/100
10/10 [==========] - 0s 4ms/step - loss: 1.0673 - accuracy: 0.7188
Epoch 25/100
10/10 [==========] - 0s 4ms/step - loss: 1.0268 - accuracy: 0.7375
41°C
Partly sunny
Search


+ Code
+ Text
Copy to Drive
Epoch 92/100
10/10 [==========] - 0s 4ms/step - loss: 0.1398 - accuracy: 0.9187
Epoch 93/100
10/10 [==========] - 0s 4ms/step - loss: 0.1377 - accuracy: 0.9187
Epoch 94/100
10/10 [==========] - 0s 6ms/step - loss: 0.1360 - accuracy: 0.9187
Epoch 95/100
10/10 [==========] - 0s 10ms/step - loss: 0.1339 - accuracy: 0.9187
Epoch 96/100
10/10 [==========] - 0s 5ms/step - loss: 0.1325 - accuracy: 0.9187
Epoch 97/100
10/10 [==========] - 0s 9ms/step - loss: 0.1313 - accuracy: 0.9187
Epoch 98/100
10/10 [==========] - 0s 5ms/step - loss: 0.1287 - accuracy: 0.9250
Epoch 99/100
10/10 [==========] - 0s 5ms/step - loss: 0.1268 - accuracy: 0.9250
Epoch 100/100
10/10 [==========] - 0s 3ms/step - loss: 0.1251 - accuracy: 0.9250
<keras.callbacks.History at 0x7fe0e81b34c0>
[ ] y_pred = ann.predict(X_test)
2/2 [==========] - 1s 7ms/step
[ ] y_pred_outputs = []
for output in y_pred:
output_list = [0] * outputs
output_list[np.where(output == max(output))[0][0]] = 1
y_pred_outputs.append(output_list)
[ ] y_pred_outputs[0]
41°C
Partly sunny
Search


+ Code
+ Text
Copy to Drive
10/10 [==========] - 0s 3ms/step - loss: 0.1251 - a
<keras.callbacks.History at 0x7fe0e81b34c0>
[ ] y_pred = ann.predict(X_test)
2/2 [==========] - 1s 7ms/step
y_pred_outputs = []
for output in y_pred:
output_list = [0] * outputs
output_list[np.where(output == max(output))[0][0]] = 1
y_pred_outputs.append(output_list)
[ ] y_pred_outputs[0]
[1, 0, 0, 0, 0]
[ ]
accuracy_score(y_test, y_pred_outputs)
0.975
[ ]
41°C
Partly sunny
Search