In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import classification_report

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
df = pd.read_csv('/content/drive/MyDrive/DL/glass.csv')
df.head()
```

Out[2]:

	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

In [3]:
```python
df.shape
```

Out[3]: (214, 10)

Exploratory Data Analysis (EDA) and Preprocessing

In [4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64  
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

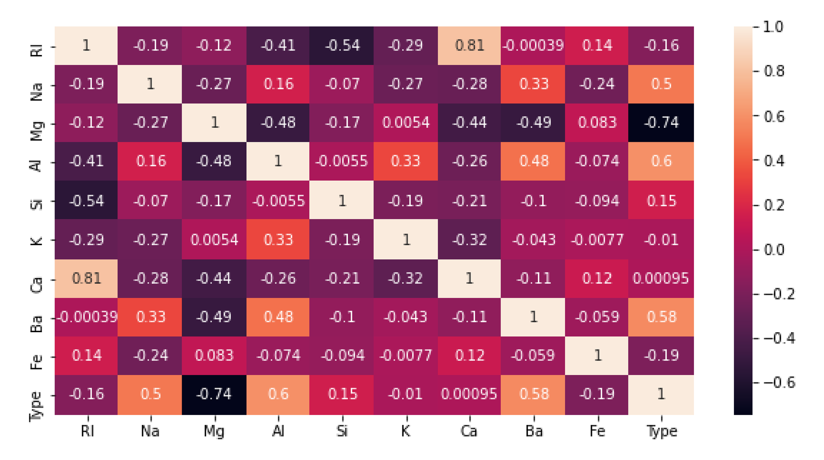
In [5]:
```python
df.describe()
```

Out[5]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Typ
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.00000
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	0.175047	0.057009	2.78037
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	0.497219	0.097439	2.10373
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.000000	0.000000	1.00000
25%	1.516523	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	0.000000	0.000000	1.00000
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	0.000000	0.000000	2.00000
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	0.000000	0.100000	3.00000
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.150000	0.510000	7.00000

In [6]:
```python
import seaborn as sns
plt.figure(figsize = (10,5))
sns.heatmap(df.corr(),annot = True)
```

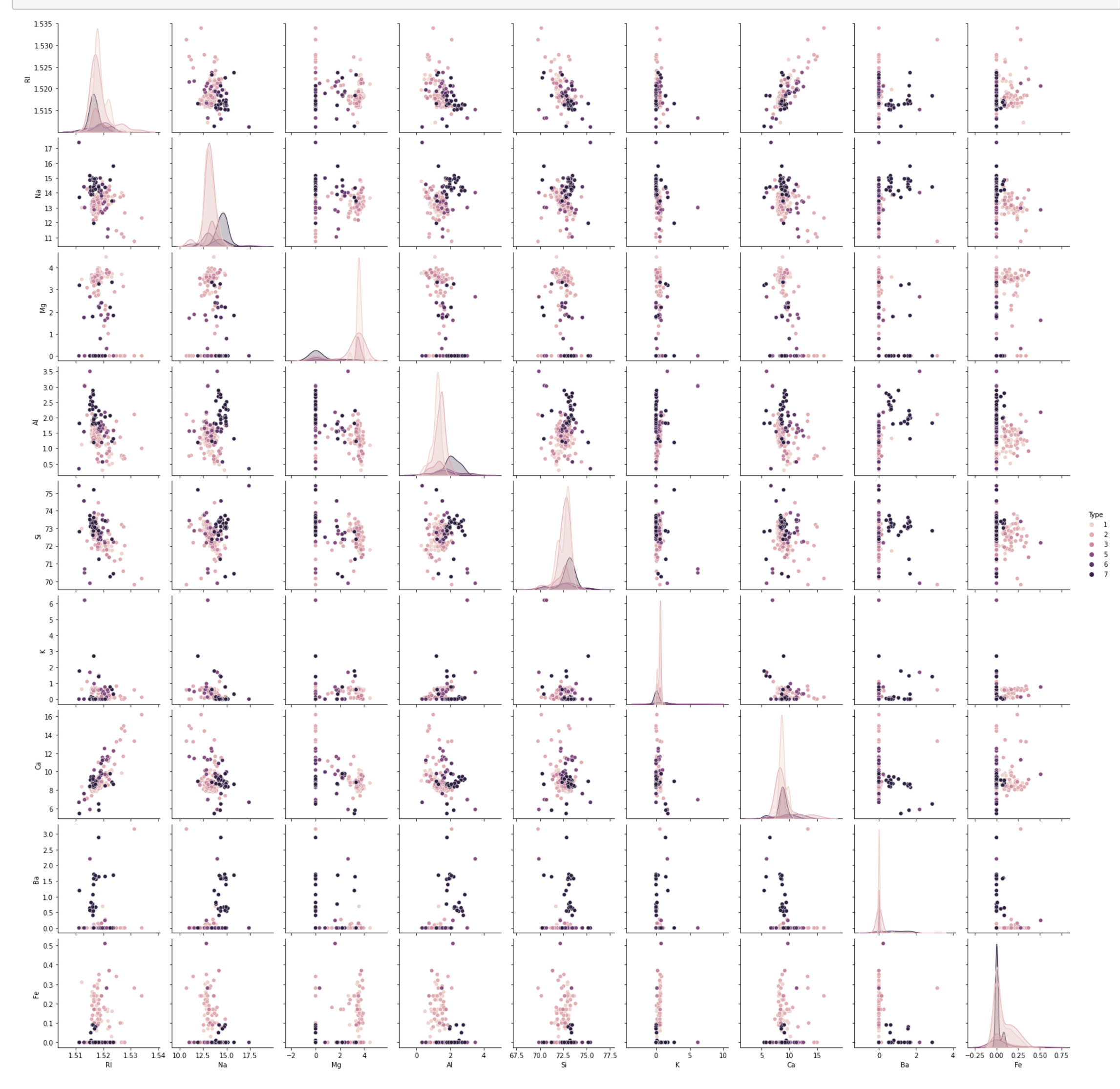
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa7e231e910>



In [7]:
```python
# value count for glass types
df.Type.value_counts()
```

Out[7]:
```
2    76
1    70
7    29
3    17
5    13
6     9
Name: Type, dtype: int64
```

In [8]:
```python
#pairwise plot of all the features
sns.pairplot(df,hue='Type')
plt.show()
```



In [9]:
```python
x = df.drop("Type",axis=1)
y = df["Type"]
```

Dividing into training and testing data

In [10]:
```python
xtrain,xtest,ytrain,ytest=train_test_split(x,y,random_state=1,test_size=0.3)
```

In [11]:
```python
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
xtrain=ss.fit_transform(xtrain)
xtest=ss.transform(xtest)
```

In [12]:
```python
model=Sequential()
model.add(Dense(64,input_dim=9,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(8,activation='softmax'))
```

In [13]:
```python
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy")
```

In [14]:
```python
trained_model=model.fit(xtrain,ytrain,epochs=50,batch_size=8)
```

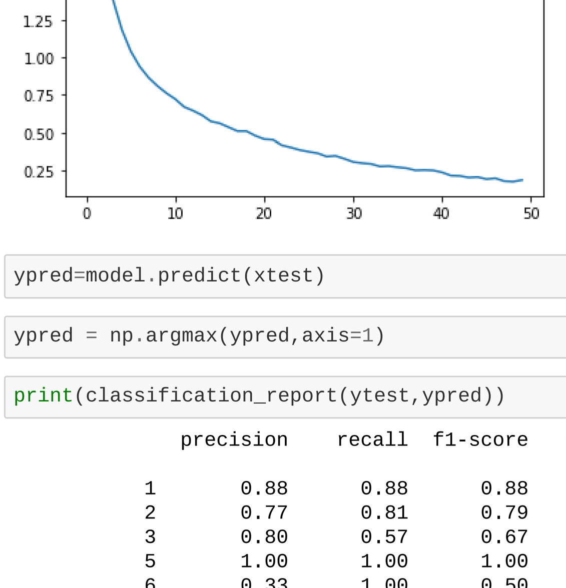
```
Epoch 1/50
19/19 [==============================] - 0s 2ms/step - loss: 2.0537
Epoch 2/50
19/19 [==============================] - 0s 2ms/step - loss: 1.8287
Epoch 3/50
19/19 [==============================] - 0s 2ms/step - loss: 1.6113
Epoch 4/50
19/19 [==============================] - 0s 1ms/step - loss: 1.3788
Epoch 5/50
19/19 [==============================] - 0s 2ms/step - loss: 1.1809
Epoch 6/50
19/19 [==============================] - 0s 1ms/step - loss: 1.0389
Epoch 7/50
19/19 [==============================] - 0s 1ms/step - loss: 0.9380
Epoch 8/50
19/19 [==============================] - 0s 2ms/step - loss: 0.8645
Epoch 9/50
19/19 [==============================] - 0s 2ms/step - loss: 0.8089
Epoch 10/50
19/19 [==============================] - 0s 2ms/step - loss: 0.7618
Epoch 11/50
19/19 [==============================] - 0s 2ms/step - loss: 0.7221
Epoch 12/50
19/19 [==============================] - 0s 2ms/step - loss: 0.6700
Epoch 13/50
19/19 [==============================] - 0s 1ms/step - loss: 0.6455
Epoch 14/50
19/19 [==============================] - 0s 1ms/step - loss: 0.6158
Epoch 15/50
19/19 [==============================] - 0s 2ms/step - loss: 0.5746
Epoch 16/50
19/19 [==============================] - 0s 2ms/step - loss: 0.5614
Epoch 17/50
19/19 [==============================] - 0s 2ms/step - loss: 0.5355
Epoch 18/50
19/19 [==============================] - 0s 1ms/step - loss: 0.5097
Epoch 19/50
19/19 [==============================] - 0s 2ms/step - loss: 0.5095
Epoch 20/50
19/19 [==============================] - 0s 2ms/step - loss: 0.4791
Epoch 21/50
19/19 [==============================] - 0s 2ms/step - loss: 0.4567
Epoch 22/50
19/19 [==============================] - 0s 2ms/step - loss: 0.4526
Epoch 23/50
19/19 [==============================] - 0s 1ms/step - loss: 0.4169
Epoch 24/50
19/19 [==============================] - 0s 1ms/step - loss: 0.4021
Epoch 25/50
19/19 [==============================] - 0s 2ms/step - loss: 0.3857
Epoch 26/50
19/19 [==============================] - 0s 1ms/step - loss: 0.3743
Epoch 27/50
19/19 [==============================] - 0s 2ms/step - loss: 0.3642
Epoch 28/50
19/19 [==============================] - 0s 1ms/step - loss: 0.3424
Epoch 29/50
19/19 [==============================] - 0s 1ms/step - loss: 0.3469
Epoch 30/50
19/19 [==============================] - 0s 1ms/step - loss: 0.3277
Epoch 31/50
19/19 [==============================] - 0s 1ms/step - loss: 0.3065
Epoch 32/50
19/19 [==============================] - 0s 1ms/step - loss: 0.2985
Epoch 33/50
19/19 [==============================] - 0s 1ms/step - loss: 0.2933
Epoch 34/50
19/19 [==============================] - 0s 1ms/step - loss: 0.2766
Epoch 35/50
19/19 [==============================] - 0s 2ms/step - loss: 0.2789
Epoch 36/50
19/19 [==============================] - 0s 1ms/step - loss: 0.2714
Epoch 37/50
19/19 [==============================] - 0s 1ms/step - loss: 0.2652
Epoch 38/50
19/19 [==============================] - 0s 1ms/step - loss: 0.2510
Epoch 39/50
19/19 [==============================] - 0s 2ms/step - loss: 0.2527
Epoch 40/50
19/19 [==============================] - 0s 1ms/step - loss: 0.2505
Epoch 41/50
19/19 [==============================] - 0s 1ms/step - loss: 0.2374
Epoch 42/50
19/19 [==============================] - 0s 2ms/step - loss: 0.2158
Epoch 43/50
19/19 [==============================] - 0s 2ms/step - loss: 0.2138
Epoch 44/50
19/19 [==============================] - 0s 2ms/step - loss: 0.2031
Epoch 45/50
19/19 [==============================] - 0s 2ms/step - loss: 0.2062
Epoch 46/50
19/19 [==============================] - 0s 2ms/step - loss: 0.1926
Epoch 47/50
19/19 [==============================] - 0s 1ms/step - loss: 0.1983
Epoch 48/50
19/19 [==============================] - 0s 2ms/step - loss: 0.1791
Epoch 49/50
19/19 [==============================] - 0s 2ms/step - loss: 0.1756
Epoch 50/50
19/19 [==============================] - 0s 1ms/step - loss: 0.1863
```

In [15]:
```python
trained_model.history['loss']
```

Out[15]:
```
[2.0536839962005615,
 1.8287111520767212,
 1.6112921237945557,
 1.3787707090377808,
 1.1808769702911377,
 1.0389350652694702,
 0.9379657506942749,
 0.8644543290138245,
 0.8089218139648438,
 0.7617931365966797,
 0.7221360206604004,
 0.6700239777565002,
 0.6455188989639282,
 0.6157588958740234,
 0.5745564699172974,
 0.5614392757415771,
 0.5354651212692261,
 0.5096593499183655,
 0.5095190405845642,
 0.4791157841682434,
 0.45671403408050537,
 0.4525596499443054,
 0.4168965518474579,
 0.4020509123802185,
 0.38566267490386963,
 0.37434816360473633,
 0.3641529083251953,
 0.34242331981658936,
 0.34687647223472595,
 0.3277430832386017,
 0.3064565658569336,
 0.29850056767463684,
 0.2933167815208435,
 0.27663886547088623,
 0.2789030075073242,
 0.27136316895484924,
 0.26517146825790405,
 0.2509898245334625,
 0.252727746963501,
 0.2504800856113434,
 0.23736341297626495,
 0.21581780910491943,
 0.21382062137126923,
 0.2031482309103012,
 0.20623844861984253,
 0.19263507425785065,
 0.19827209413051605,
 0.1791045069694519,
 0.17556676268577576,
 0.18631522357463837]
```

In [16]:
```python
plt.plot(trained_model.history['loss'])
```

Out[16]: [<matplotlib.lines.Line2D at 0x7fa7cd89a610>]



In [17]:
```python
ypred=model.predict(xtest)
```

In [18]:
```python
ypred = np.argmax(ypred,axis=1)
```

In [19]:
```python
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           1       0.88      0.88      0.88        25
           2       0.77      0.81      0.79        21
           3       0.80      0.57      0.67         7
           5       1.00      1.00      1.00         2
           6       0.33      1.00      0.50         1
           7       1.00      0.89      0.94         9

    accuracy                           0.83        65
   macro avg       0.80      0.86      0.80        65
weighted avg       0.85      0.83      0.83        65
```