Predicting Wine Types

Importing the dataset

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
```

[1066] ✓ 0.2s

Data Preprocessing

We will assign 1 for red wine and 0 for white wine.

```python
dataset = pd.read_csv("data/wine_dataset.csv")
dataset['style'] = dataset['style'].replace('red', 1)
dataset['style'] = dataset['style'].replace('white', 0)
df = pd.DataFrame(dataset)
dataset.head()
```

[1067] ✓ 0.6s

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality	style
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	1
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	1
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	1
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	1

Data Analysis and Visualization

```python
print(df[df.columns[0:12]])
```

[1068] ✓ 0.5s

```
Output exceeds the size limit. Open the full output data in a text editor
      fixed_acidity  volatile_acidity  citric_acid  residual_sugar  chlorides  \
0               7.4              0.70         0.00             1.9      0.076
1               7.8              0.88         0.00             2.6      0.098
2               7.8              0.76         0.04             2.3      0.092
3              11.2              0.28         0.56             1.9      0.075
4               7.4              0.70         0.00             1.9      0.076
...             ...               ...          ...             ...        ...
6492            6.2              0.21         0.29             1.6      0.039
6493            6.6              0.32         0.36             8.0      0.047
6494            6.5              0.24         0.19             1.2      0.041
6495            5.5              0.29         0.30             1.1      0.022
6496            6.0              0.21         0.38             0.8      0.020

      free_sulfur_dioxide  total_sulfur_dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
...                   ...                   ...      ...   ...        ...
6492                 24.0                  92.0  0.99114  3.27       0.50
6493                 57.0                 168.0  0.99490  3.15       0.46
6494                 30.0                 111.0  0.99254  2.99       0.46
6495                 20.0                 110.0  0.98869  3.34       0.38
6496                 22.0                  98.0  0.98941  3.26       0.32
...
6495     12.8        7
6496     11.8        6

[6497 rows x 12 columns]
```

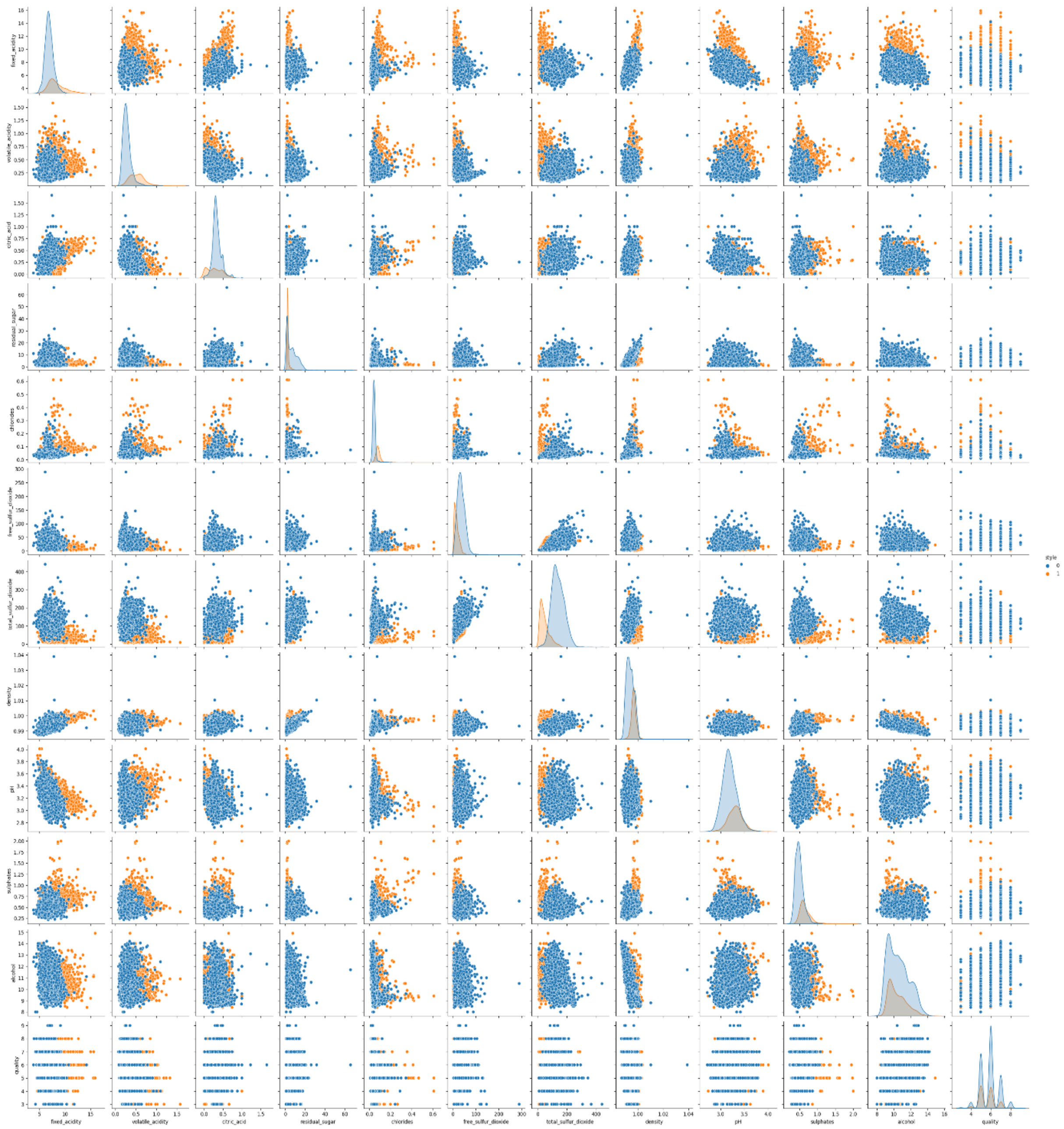
```python
dataset.describe()
```

✓ 0.7s

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality	style
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.744574	0.994697	3.218501	0.531268	10.491801	5.818378	0.246114
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806	1.192712	0.873255	0.430779
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000	3.000000	0.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000	9.500000	5.000000	0.000000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10.300000	6.000000	0.000000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000	11.300000	6.000000	0.000000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000	9.000000	1.000000

```python
sns.pairplot(df,hue='style')
```

✓ 1m 52.6s

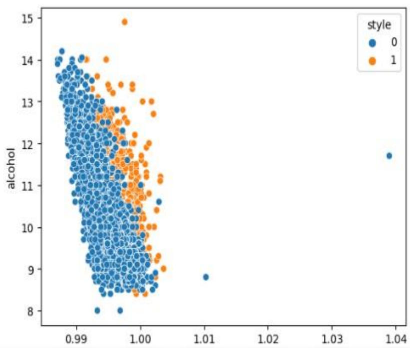

fixed_acidity
volatile_acidity
citric_acid
residual_sugar
chlorides
free_sulfur_dioxide
total_sulfur_dioxide
density
pH
sulphates
alcohol
quality
style
0
1

```python
corr = df.corr()
corr.style.background_gradient()
```
[1071] ✓ 0.7s

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality	style
fixed_acidity	1.000000	0.219008	0.324436	-0.111981	0.298195	-0.282735	-0.329054	0.458910	-0.252700	0.299568	-0.095452	-0.076743	0.486740
volatile_acidity	0.219008	1.000000	-0.377981	-0.196011	0.377124	-0.352557	-0.414476	0.271296	0.261454	0.225984	-0.037640	-0.265699	0.653036
citric_acid	0.324436	-0.377981	1.000000	0.142451	0.038998	0.133126	0.195242	0.096154	-0.329808	0.056197	-0.010493	0.085532	-0.187397
residual_sugar	-0.111981	-0.196011	0.142451	1.000000	-0.128940	0.402871	0.495482	0.552517	-0.267320	-0.185927	-0.359415	-0.036980	-0.348821
chlorides	0.298195	0.377124	0.038998	-0.128940	1.000000	-0.195045	-0.279630	0.362615	0.044708	0.395593	-0.256916	-0.200666	0.512678
free_sulfur_dioxide	-0.282735	-0.352557	0.133126	0.402871	-0.195045	1.000000	0.720934	0.025717	-0.145854	-0.188457	-0.179838	0.055463	-0.471644
total_sulfur_dioxide	-0.329054	-0.414476	0.195242	0.495482	-0.279630	0.720934	1.000000	0.032395	-0.238413	-0.275727	-0.265740	-0.041385	-0.700357
density	0.458910	0.271296	0.096154	0.552517	0.362615	0.025717	0.032395	1.000000	0.011686	0.259478	-0.686745	-0.305858	0.390645
pH	-0.252700	0.261454	-0.329808	-0.267320	0.044708	-0.145854	-0.238413	0.011686	1.000000	0.192123	0.121248	0.019506	0.329129
sulphates	0.299568	0.225984	0.056197	-0.185927	0.395593	-0.188457	-0.275727	0.259478	0.192123	1.000000	-0.003029	0.038485	0.487218
alcohol	-0.095452	-0.037640	-0.010493	-0.359415	-0.256916	-0.179838	-0.265740	-0.686745	0.121248	-0.003029	1.000000	0.444319	-0.032970
quality	-0.076743	-0.265699	0.085532	-0.036980	-0.200666	0.055463	-0.041385	-0.305858	0.019506	0.038485	0.444319	1.000000	-0.119323
style	0.486740	0.653036	-0.187397	-0.348821	0.512678	-0.471644	-0.700357	0.390645	0.329129	0.487218	-0.032970	-0.119323	1.000000

Rq: density and alcohol are the most correlated features

```python
sns.scatterplot(data=df, x='density', y='alcohol', hue='style')
plt.show()
```
[1072] ✓ 0.4s




```python
X = dataset.values[:,range(0,12)]
y = dataset.values[:,range(12,13)]
print(X)
print(y)
```
[1073] ✓ 0.4s

```
[[ 7.4   0.7   0.   ... 0.56  9.4   5.  ]
 [ 7.8   0.88  0.   ... 0.68  9.8   5.  ]
 [ 7.8   0.76  0.04 ... 0.65  9.8   5.  ]
 ...
 [ 6.5   0.24  0.19 ... 0.46  9.4   6.  ]
 [ 5.5   0.29  0.3  ... 0.38 12.8   7.  ]
 [ 6.    0.21  0.38 ... 0.32 11.8   6.  ]]
[[1.]
 [1.]
 [1.]
 ...
 [0.]
 [0.]
 [0.]]
```

```python
print('dimensions de X:', X.shape)
print('dimensions de Y:', y.shape)
```
[1074] ✓ 0.3s

```
dimensions de X: (6497, 12)
dimensions de Y: (6497, 1)
```

Preparing the model

```
def initialisation(X):
    W = np.random.randn(X.shape[1], 1)
    b = np.random.randn(1)
    return (W, b)
```

[1075] ✓ 0.3s

```
initialisation(X)
```

[1076] ✓ 0.3s

```
(array([[ 1.15189222],
        [ 0.42954223],
        [ 0.3865618 ],
        [ 0.88198986],
        [ 0.8173989 ],
        [ 0.88860612],
        [-0.88150615],
        [-1.44645573],
        [ 0.14764371],
        [-0.13950698],
        [-0.71451461],
        [ 1.10652459]]),
 array([-1.0007181]))
```

```
def model(X, W, b):
    Z = X.dot(W) + b
    A = 1 / (1 + np.exp(-Z))
    return A
```

[1077] ✓ 0.3s

```
W,b = initialisation(X)
model(X,W,b)
```

[1078] ✓ 0.4s

```
array([[4.17096197e-20],
       [2.84536084e-36],
       [1.93376608e-30],
       ...,
       [2.80469319e-59],
       [4.32385208e-61],
       [2.10513749e-53]])
```

```
def log_loss(A, y):
    return 1 / len(y) * np.sum(-y * np.log(A) - (1 - y) * np.log(1 - A))
```

[1079] ✓ 0.3s

```
def gradients(A, X, y):
    dW = 1 / len(y) * np.dot(X.T, A - y)
    db = 1 / len(y) * np.sum(A - y)
    return (dW, db)
```

[1080] ✓ 0.3s

```
def update(dW, db, W, b, learning_rate):
    W = W - learning_rate * dW
    b = b - learning_rate * db
    return (W, b)
```

[1081] ✓ 0.3s

```
def predict(X, W, b):
    A = model(X, W, b)
    # print(A)
    return A >= 0.5
```

[1082] ✓ 0.4s

```python
def update(dW, db, W, b, learning_rate):
    W = W - learning_rate * dW
    b = b - learning_rate * db
    return (W, b)
```

[1081] ✓ 0.3s

```python
def predict(X, W, b):
    A = model(X, W, b)
    # print(A)
    return A >= 0.5
```

[1082] ✓ 0.4s

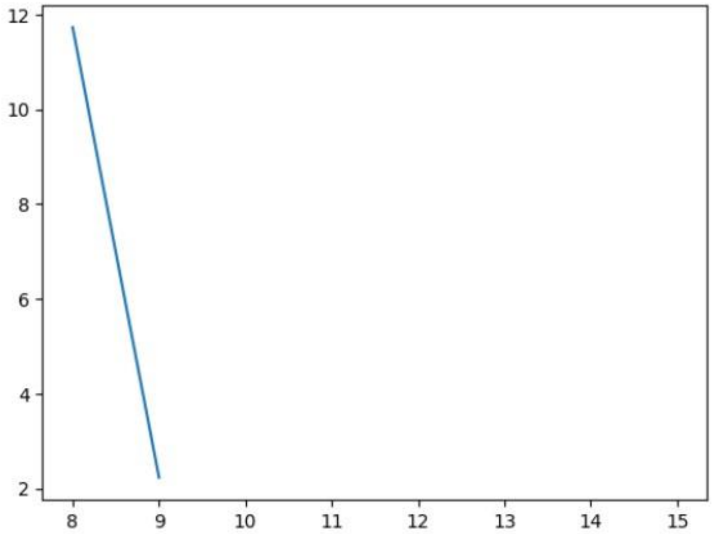
```python
def artificial_neuron(X, y, learning_rate = 0.1, n_iter = 100):
    W, b = initialisation(X)

    Loss = []

    for i in range(n_iter):
        A = model(X, W, b)
        Loss.append(log_loss(A, y))
        dW, db = gradients(A, X, y)
        W, b = update(dW, db, W, b, learning_rate)

    y_pred = predict(X, W, b)

    plt.plot(Loss)

    print("Accuracy: ",accuracy_score(y, y_pred))
    print("Recall: ",recall_score(y, y_pred))
    print("Precision: ",precision_score(y, y_pred))


    return (W, b)
```

[1083] ✓ 0.3s

```python
W, b = artificial_neuron(X,y)
```

[1084] ✓ 0.2s

```
Accuracy:  0.8123749422810528
Recall:  0.9224515322076298
Precision:  0.5739299610894941
```



```python
new_wine=np.array([8.9, 0.22, 0.48, 1.8, 0.0077, 29.0, 60.0, 0.9968,3.39, 0.53, 9.4, 6.0]) # white_wine
new_wine2=np.array([8.9,0.62,0.19,3.9,0.17,51,148,0.9986,3.17,0.93,9.2,5]) # red_wine
res = predict(new_wine2, W ,b)

if res :
    print('The type of this wine is white')
else:
    print('The type of this wine is red')
```

085] ✓ 0.3s

```
The type of this wine is red
```