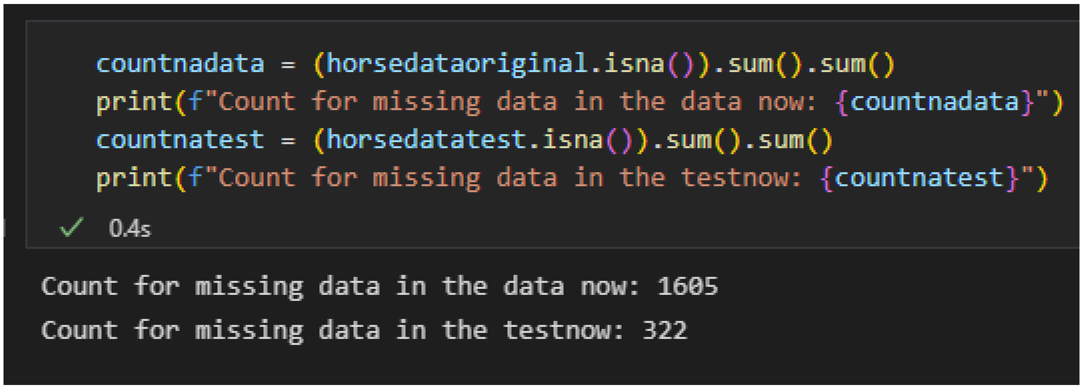
Steven Jonathan 溫志恩

STUDENT ID : 410821322

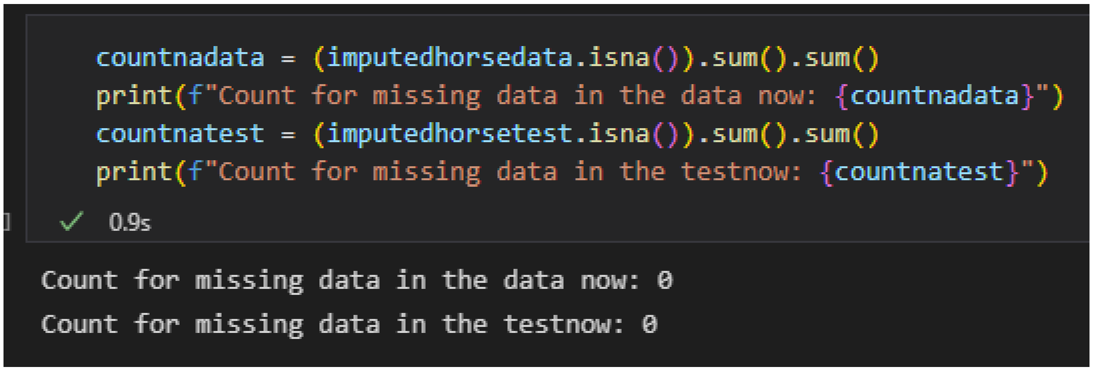
Before Imputing - A lot of NaN values on files .data and .test



```
countnadata = (horsedataoriginal.isna()).sum().sum()
print(f"Count for missing data in the data now: {countnadata}")
countnatest = (horsedatatest.isna()).sum().sum()
print(f"Count for missing data in the testnow: {countnatest}")
```

```
Count for missing data in the data now: 1605
Count for missing data in the testnow: 322
```

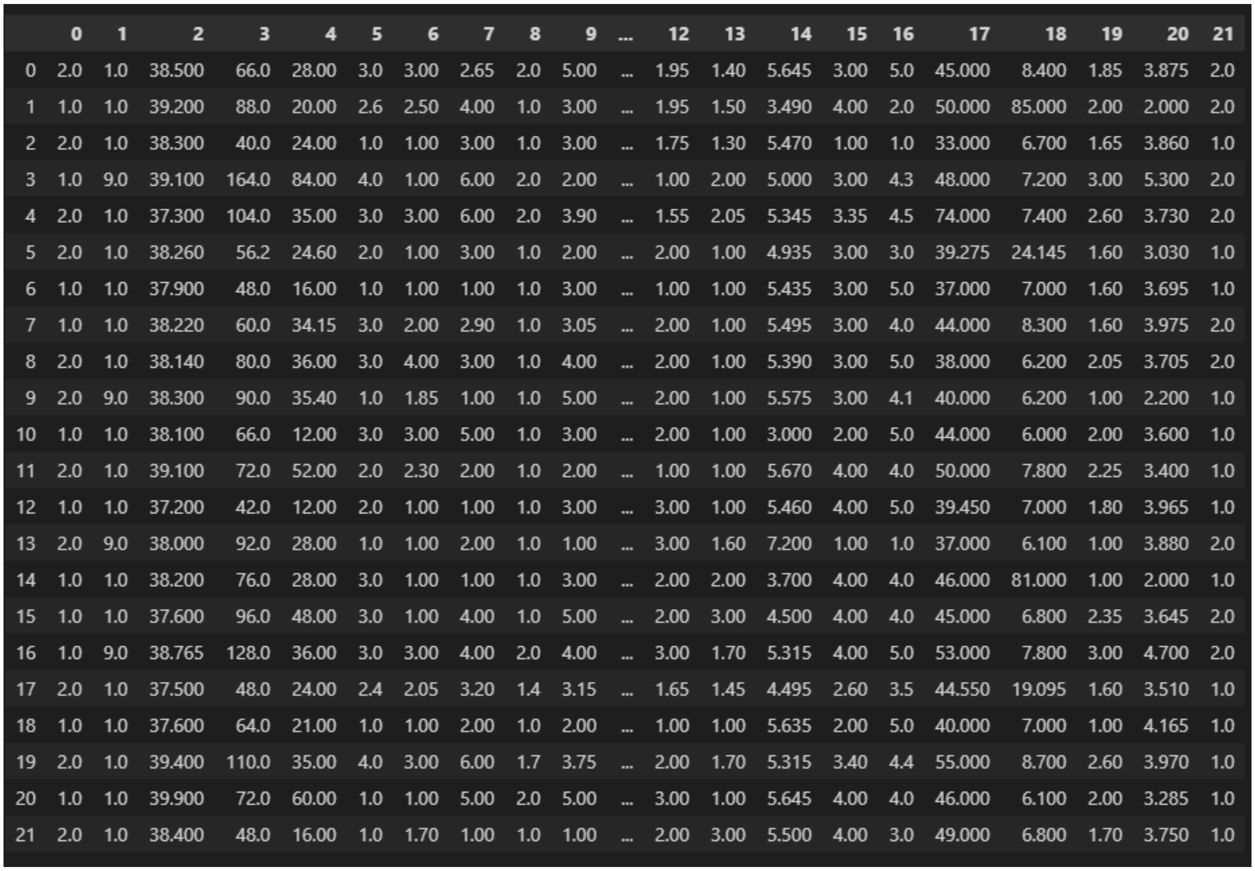
After imputing - All NaN values are gone



```
countnadata = (imputedhorsedata.isna()).sum().sum()
print(f"Count for missing data in the data now: {countnadata}")
countnatest = (imputedhorsetest.isna()).sum().sum()
print(f"Count for missing data in the testnow: {countnatest}")
```

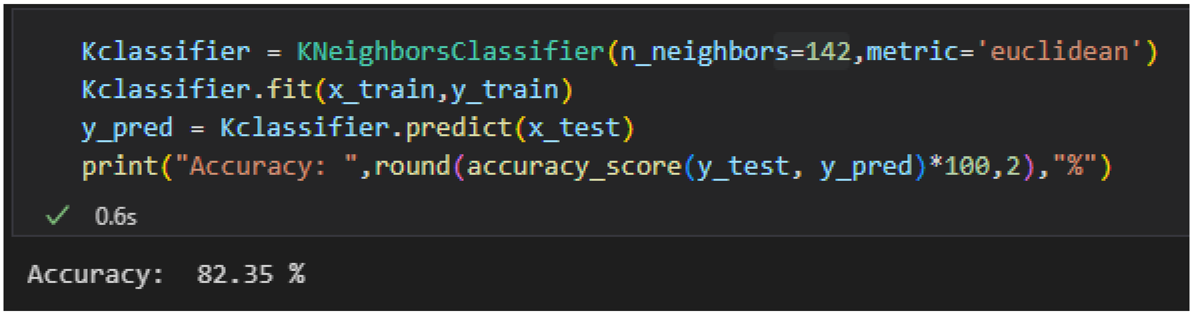
```
Count for missing data in the data now: 0
Count for missing data in the testnow: 0
```

.data after imputation



	0	1	2	3	4	5	6	7	8	9	...	12	13	14	15	16	17	18	19	20	21
0	2.0	1.0	38.500	66.0	28.00	3.0	3.00	2.65	2.0	5.00	...	1.95	1.40	5.645	3.00	5.0	45.000	8.400	1.85	3.875	2.0
1	1.0	1.0	39.200	88.0	20.00	2.6	2.50	4.00	1.0	3.00	...	1.95	1.50	3.490	4.00	2.0	50.000	85.000	2.00	2.000	2.0
2	2.0	1.0	38.300	40.0	24.00	1.0	1.00	3.00	1.0	3.00	...	1.75	1.30	5.470	1.00	1.0	33.000	6.700	1.65	3.860	1.0
3	1.0	9.0	39.100	164.0	84.00	4.0	1.00	6.00	2.0	2.00	...	1.00	2.00	5.000	3.00	4.3	48.000	7.200	3.00	5.300	2.0
4	2.0	1.0	37.300	104.0	35.00	3.0	3.00	6.00	2.0	3.90	...	1.55	2.05	5.345	3.35	4.5	74.000	7.400	2.60	3.730	2.0
5	2.0	1.0	38.260	56.2	24.60	2.0	1.00	3.00	1.0	2.00	...	2.00	1.00	4.935	3.00	3.0	39.275	24.145	1.60	3.030	1.0
6	1.0	1.0	37.900	48.0	16.00	1.0	1.00	1.00	1.0	3.00	...	1.00	1.00	5.435	3.00	5.0	37.000	7.000	1.60	3.695	1.0
7	1.0	1.0	38.220	60.0	34.15	3.0	2.00	2.90	1.0	3.05	...	2.00	1.00	5.495	3.00	4.0	44.000	8.300	1.60	3.975	2.0
8	2.0	1.0	38.140	80.0	36.00	3.0	4.00	3.00	1.0	4.00	...	2.00	1.00	5.390	3.00	5.0	38.000	6.200	2.05	3.705	2.0
9	2.0	9.0	38.300	90.0	35.40	1.0	1.85	1.00	1.0	5.00	...	2.00	1.00	5.575	3.00	4.1	40.000	6.200	1.00	2.200	1.0
10	1.0	1.0	38.100	66.0	12.00	3.0	3.00	5.00	1.0	3.00	...	2.00	1.00	3.000	2.00	5.0	44.000	6.000	2.00	3.600	1.0
11	2.0	1.0	39.100	72.0	52.00	2.0	2.30	2.00	1.0	2.00	...	1.00	1.00	5.670	4.00	4.0	50.000	7.800	2.25	3.400	1.0
12	1.0	1.0	37.200	42.0	12.00	2.0	1.00	1.00	1.0	3.00	...	3.00	1.00	5.460	4.00	5.0	39.450	7.000	1.80	3.965	1.0
13	2.0	9.0	38.000	92.0	28.00	1.0	1.00	2.00	1.0	1.00	...	3.00	1.60	7.200	1.00	1.0	37.000	6.100	1.00	3.880	2.0
14	1.0	1.0	38.200	76.0	28.00	3.0	1.00	1.00	1.0	3.00	...	2.00	2.00	3.700	4.00	4.0	46.000	81.000	1.00	2.000	1.0
15	1.0	1.0	37.600	96.0	48.00	3.0	1.00	4.00	1.0	5.00	...	2.00	3.00	4.500	4.00	4.0	45.000	6.800	2.35	3.645	2.0
16	1.0	9.0	38.765	128.0	36.00	3.0	3.00	4.00	2.0	4.00	...	3.00	1.70	5.315	4.00	5.0	53.000	7.800	3.00	4.700	2.0
17	2.0	1.0	37.500	48.0	24.00	2.4	2.05	3.20	1.4	3.15	...	1.65	1.45	4.495	2.60	3.5	44.550	19.095	1.60	3.510	1.0
18	1.0	1.0	37.600	64.0	21.00	1.0	1.00	2.00	1.0	2.00	...	1.00	1.00	5.635	2.00	5.0	40.000	7.000	1.00	4.165	1.0
19	2.0	1.0	39.400	110.0	35.00	4.0	3.00	6.00	1.7	3.75	...	2.00	1.70	5.315	3.40	4.4	55.000	8.700	2.60	3.970	1.0
20	1.0	1.0	39.900	72.0	60.00	1.0	1.00	5.00	2.0	5.00	...	3.00	1.00	5.645	4.00	4.0	46.000	6.100	2.00	3.285	1.0
21	2.0	1.0	38.400	48.0	16.00	1.0	1.70	1.00	1.0	1.00	...	2.00	3.00	5.500	4.00	3.0	49.000	6.800	1.70	3.750	1.0

KNeighborsClassifier



```
Kclassifier = KNeighborsClassifier(n_neighbors=142,metric='euclidean')
Kclassifier.fit(x_train,y_train)
y_pred = Kclassifier.predict(x_test)
print("Accuracy: ",round(accuracy_score(y_test, y_pred)*100,2),"%")
```

✓ 0.6s

```
Accuracy:  82.35 %
```