



L OVELY
P ROFESSIONAL
U NIVERSITY

Name : Raviteja Gopavaram

Roll Number: A06

Reg Number: 11804302

Section: KM007

Project Name: Artificial Characters

The screenshot displays two instances of Microsoft Excel, each showing a different worksheet from the same workbook named "phpQRHPH.csv".

Top Worksheet:

V1	V2	V3	V4	V5	V6	V7	Class
0	0	0	0	20	20	46.1	1
1	19	0	19	8	8	46.1	1
2	0	20	19	8	22.47	46.1	1
3	0	20	8	42	23.41	46.1	1
4	19	8	8	42	35.74	46.1	1
0	0	0	0	12	12	37.66	1
1	7	0	7	7	7	37.66	1
2	0	12	7	7	8.6	37.66	1
3	0	12	1	35	23.02	37.66	1
4	7	7	5	30	23.09	37.66	1
5	1	35	2	37	2.24	37.66	1
6	2	37	5	30	7.62	37.66	1
0	0	0	0	20	20	43.05	1
1	19	0	20	11	11.05	43.05	1
2	0	20	20	11	21.93	43.05	1
3	0	20	0	37	17	43.05	1
4	20	11	22	22	11.18	43.05	1
5	0	37	22	22	26.63	43.05	1
0	0	0	0	7	7	25	1
1	6	0	6	7	7	25	1
2	0	7	6	7	6	25	1
3	0	7	1	24	17.03	25	1
4	6	7	7	24	17.03	25	1
5	1	24	2	24	1	25	1
6	2	24	7	24	5	25	1
0	0	0	0	7	7	40	1
1	21	0	22	11	11.05	40	1
2	0	7	22	11	22.36	40	1
3	0	7	2	15	8.25	40	1

Bottom Worksheet:

A	B	C	D	E	F	G	H
4	22	11	24	32	21.1	40	1
5	2	15	24	32	27.8	40	1
0	0	0	0	6	6	26	1
1	10	0	10	7	7	26	1
2	0	6	10	7	10.05	26	1
3	0	6	1	13	7.07	26	1
4	10	7	9	24	17.03	26	1
5	1	13	5	22	9.85	26	1
6	5	22	9	24	4.47	26	1
0	0	0	0	22	22	41.59	1
1	18	0	19	11	11.05	41.59	1
2	0	22	19	11	21.95	41.59	1
3	0	22	1	37	15.03	41.59	1
4	19	11	17	30	19.1	41.59	1
5	1	37	17	30	17.46	41.59	1
0	0	0	0	24	24	49.74	1
1	24	0	25	17	17.03	49.74	1
2	0	24	25	17	25.96	49.74	1
3	0	24	2	34	10.2	49.74	1
4	25	17	24	40	23.02	49.74	1
5	2	34	9	43	11.4	49.74	1
6	9	43	24	40	15.3	49.74	1
0	0	0	0	10	10	32.53	1
1	23	0	23	14	14	32.53	1
2	0	10	23	14	23.35	32.53	1
3	0	10	13	23	18.38	32.53	1
4	23	14	13	23	13.45	32.53	1
0	0	0	0	15	15	37.48	1
1	25	0	27	9	9.22	37.48	1
2	0	15	27	9	27.66	37.48	1

Basic Libraries for the Data And Exploring the Data

```
[1] import pandas as pd
import numpy as np
from google.colab import files
ex=files.upload()
```

	0	1	2	3	4	5	6	7
0	V1	V2	V3	V4	V5	V6	V7	Class
1	0	0	0	0	20	20	46.1	1
2	1	19	0	19	8	8	46.1	1
3	2	0	20	19	8	22.47	46.1	1
4	3	0	20	8	42	23.41	46.1	1

Understanding the Data

```
[28] df.dtypes
```

```
0    int64
1    int64
2    int64
3    int64
4    int64
5    int64
6    int64
7    int64
dtype: object
```



```
df.shape
```

```
(10219, 8)
```



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



```
0    0
```

```
1    0
```

```
2    0
```

```
3    0
```

```
4    0
```

```
5    0
```

```
6    0
```

```
7    0
```

```
dtype: int64
```



df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10219 entries, 0 to 10218
Data columns (total 8 columns):
#   Column  Non-Null Count  Dtype
---  -
0   0        10219 non-null  object
1   1        10219 non-null  object
2   2        10219 non-null  object
3   3        10219 non-null  object
4   4        10219 non-null  object
5   5        10219 non-null  object
6   6        10219 non-null  object
7   7        10219 non-null  object
dtypes: object(8)
memory usage: 638.8+ KB
```


```
[7] from sklearn.preprocessing import LabelEncoder
     le=LabelEncoder()
     for i in df:
         df[i]=le.fit_transform(df[i])
```


▶ df.info()

```
↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 10219 entries, 0 to 10218
Data columns (total 8 columns):
#   Column  Non-Null Count  Dtype
---  -
0   0        10219 non-null  int64
1   1        10219 non-null  int64
2   2        10219 non-null  int64
3   3        10219 non-null  int64
4   4        10219 non-null  int64
5   5        10219 non-null  int64
6   6        10219 non-null  int64
7   7        10219 non-null  int64
dtypes: int64(8)
memory usage: 638.8 KB
```

Importing LabelEncoder

Prepared Data

 `data.head()`



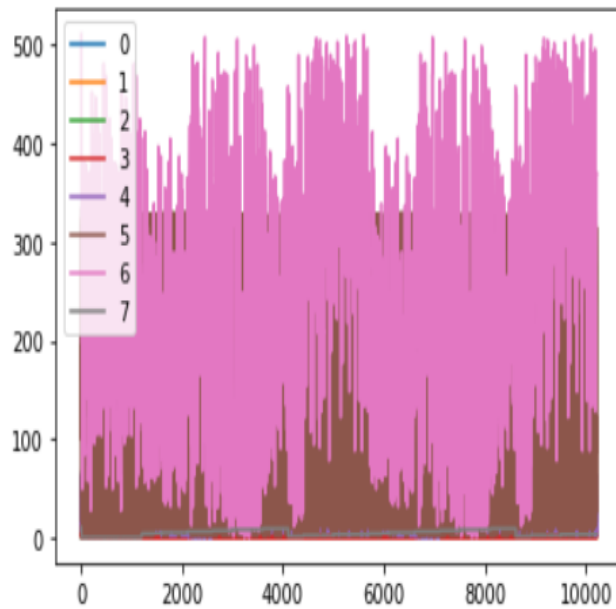
	0	1	2	3	4	5	6
0	8	45	63	48	66	333	511
1	0	0	0	0	15	101	325
2	1	11	0	11	64	319	325
3	2	0	12	11	64	129	325
4	3	0	12	46	39	141	325

Data Visualization



```
df.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3389f2fa10>
```



Scaling Data

```
[15] from sklearn.model_selection import train_test_split
      x_train, x_test, y_train, y_test=train_test_split(data, target, test_size=0.3,random_state=42)
```

```
[16] from sklearn.preprocessing import StandardScaler
      sc=StandardScaler()
      x_train_sc=sc.fit_transform(x_train)
      x_test_sc=sc.fit_transform(x_test)
```

```
[17] x_train_sc=pd.DataFrame(x_train_sc)
      x_test_sc=pd.DataFrame(x_test_sc)
```

```
[18] x_train_sc.describe().round(2)
```

	0	1	2	3	4	5	6
count	7153.00	7153.00	7153.00	7153.00	7153.00	7153.00	7153.00
mean	-0.00	0.00	-0.00	-0.00	0.00	-0.00	-0.00
std	1.00	1.00	1.00	1.00	1.00	1.00	1.00
min	-1.29	-0.60	-0.88	-0.79	-1.37	-1.28	-1.77
25%	-0.71	-0.60	-0.88	-0.79	-0.77	-0.92	-0.83
50%	-0.13	-0.60	-0.36	-0.36	-0.23	-0.28	-0.07

Importing SVC and Predicting

```
▶ from sklearn.svm import SVC  
svc=SVC()  
svc.fit(x_train, y_train)
```

```
↳ SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,  
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',  
    max_iter=-1, probability=False, random_state=None, shrinking=True,  
    tol=0.001, verbose=False)
```

[+ Code](#)[+ Text](#)

```
[20] pred_train=svc.predict(x_train)
```

```
[21] from sklearn.metrics import accuracy_score  
print(accuracy_score(pred_train, y_train))
```

```
0.3124563120369076
```

```
[22] pred_test=svc.predict(x_test)  
print(accuracy_score(pred_test, y_test))
```

```
0.2762557077625571
```

Importing Accuracy_score and Accurating the Data

```
[21] from sklearn.metrics import accuracy_score  
      print(accuracy_score(pred_train, y_train))
```

0.3124563120369076

```
[22] pred_test=svc.predict(x_test)  
      print(accuracy_score(pred_test, y_test))
```

0.2762557077625571

```
[23] from sklearn.naive_bayes import GaussianNB  
      nb=GaussianNB()  
      nb.fit(x_train, y_train)  
      pred_train=nb.predict(x_train)  
      pred_test=nb.predict(x_test)  
      from sklearn.metrics import accuracy_score  
      print(accuracy_score(pred_train, y_train))  
      print(accuracy_score(pred_test, y_test))
```

0.21277785544526773

0.2146118721461187

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier(criterion='entropy', max_depth=7)
dt.fit(x_train, y_train)
pred_train_dt=dt.predict(x_train)
pred_test_dt=dt.predict(x_test)
print(accuracy_score(pred_train_dt, y_train))
print(accuracy_score(pred_test_dt, y_test))
```

0.5362784845519363

0.5068493150684932



Bagging Classifier

```
▶ from sklearn.ensemble import BaggingClassifier  
  bag=BaggingClassifier(n_estimators=10)  
  bag.fit(x_train_sc, y_train)  
  print(accuracy_score(bag.predict(x_train_sc), y_train))  
  print(accuracy_score(bag.predict(x_test_sc), y_test))
```

```
↳ 0.986579057738012  
   0.8118069145466406
```

Random Forest Classifier

```
[26] from sklearn.ensemble import RandomForestClassifier
      rfc=RandomForestClassifier(n_estimators=10)
      rfc.fit(x_train_sc, y_train)
      print(accuracy_score(rfc.predict(x_train_sc), y_train))
      print(accuracy_score(rfc.predict(x_test_sc), y_test))
```

```
0.9893750873759262
0.8255055446836269
```

```
▶ from sklearn.ensemble import RandomForestClassifier
   rfc=RandomForestClassifier(n_estimators=10)
   rfc.fit(x_train_sc, y_train)
   print(accuracy_score(rfc.predict(x_train_sc), y_train))
   print(accuracy_score(rfc.predict(x_test_sc), y_test))
```

```
↗ 0.9886760799664477
   0.8398564905414221
```