hw7

March 9, 2024

[1]:
```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from IPython.display import display, HTML
```

[2]:
```
import warnings
warnings.filterwarnings("ignore")
```

[3]:
```
# reading the data
df = pd.read_csv('malware_multiclass.csv')
print(df.shape)
df=df.rename(columns=lambda x: x.strip())
cols=df.columns
display(HTML(df.head(10).to_html()))
```

```
(100000, 36)
```

```
<IPython.core.display.HTML object>
```

[4]:
```
# selecting columns specified in readme file
df = df.loc[:, ['classification', 'os', 'usage_counter', 'prio', 'static_prio',␣
 ↪'normal_prio', 'vm_pgoff', 'vm_truncate_count', 'task_size', 'map_count',␣
 ↪'hiwater_rss', 'total_vm', 'shared_vm', 'exec_vm', 'reserved_vm', 'nr_ptes',␣
 ↪'nvcsw', 'nivcsw', 'signal_nvcsw']]
display(HTML(df.head(10).to_html()))
```

```
<IPython.core.display.HTML object>
```

[5]:
```
# checking if the data has null values
df.isna().sum()
```

[5]:
```
classification       0
os                   0
usage_counter        0
prio                 0
static_prio          0
normal_prio          0
vm_pgoff             0
```

```
vm_truncate_count    0
task_size            0
map_count            0
hiwater_rss          0
total_vm             0
shared_vm            0
exec_vm              0
reserved_vm          0
nr_ptes              0
nvcsw                0
nivcsw               0
signal_nvcsw         0
dtype: int64
```

[ ]:

[6]:
```
# encode labels
y = df['classification'] # define label as nominal values
le = preprocessing.LabelEncoder()
le.fit(y)
y_encoded = le.transform(y) # encode nominal labels to integers␣
 ↪#####################################
df['classification'] = y_encoded
```

[ ]:

[7]:
```
# print out and display dataframe as tables in HTML
display(HTML(df.head(10).to_html()))
```

```
<IPython.core.display.HTML object>
```

[8]:
```
# convert all nominal variables to binary variables
df_num=df.copy(deep=True)
df_dummies=pd.get_dummies(df_num['os'],dtype=float)
df_num=df_num.join(df_dummies)
# drop original columns
df_num=df_num.drop('os',axis=1)
df_num.head()
```

[8]:

	classification	usage_counter	prio	static_prio	normal_prio	\
0	1	0	3069378560	14274	0	
1	1	0	3069378560	14274	0	
2	1	0	3069378560	14274	0	
3	1	0	3069378560	14274	0	
4	1	0	3069378560	14274	0	

	vm_pgoff	vm_truncate_count	task_size	map_count	hiwater_rss	…	\

```
0         0              13173          0       6850            0  …
1         0              13173          0       6850            0  …
2         0              13173          0       6850            0  …
3         0              13173          0       6850            0  …
4         0              13173          0       6850            0  …

   reserved_vm  nr_ptes   nvcsw  nivcsw  signal_nvcsw  CentOS  Debian  Mac  \
0          210        0  341974       0             0     1.0     0.0  0.0
1          210        0  341974       0             0     0.0     0.0  0.0
2          210        0  341974       0             0     0.0     0.0  1.0
3          210        0  341974       0             0     0.0     0.0  0.0
4          210        0  341974       0             0     0.0     0.0  1.0

   Ubuntu  Windows
0     0.0      0.0
1     0.0      1.0
2     0.0      0.0
3     1.0      0.0
4     0.0      0.0

[5 rows x 23 columns]
```

[9]:
```
# dropping the extra column
df_num.drop('CentOS', axis=1, inplace=True)
df_num.head()
```

[9]:
```
   classification  usage_counter        prio  static_prio  normal_prio  \
0               1              0  3069378560        14274            0
1               1              0  3069378560        14274            0
2               1              0  3069378560        14274            0
3               1              0  3069378560        14274            0
4               1              0  3069378560        14274            0

   vm_pgoff  vm_truncate_count  task_size  map_count  hiwater_rss  …  \
0         0              13173          0       6850            0  …
1         0              13173          0       6850            0  …
2         0              13173          0       6850            0  …
3         0              13173          0       6850            0  …
4         0              13173          0       6850            0  …

   exec_vm  reserved_vm  nr_ptes   nvcsw  nivcsw  signal_nvcsw  Debian  Mac  \
0      124          210        0  341974       0             0     0.0  0.0
1      124          210        0  341974       0             0     0.0  0.0
2      124          210        0  341974       0             0     0.0  1.0
3      124          210        0  341974       0             0     0.0  0.0
4      124          210        0  341974       0             0     0.0  1.0
```

```
   Ubuntu  Windows
0     0.0      0.0
1     0.0      1.0
2     0.0      0.0
3     1.0      0.0
4     0.0      0.0

[5 rows x 22 columns]
```

[10]:
```
df_num.head()
```

[10]:
```
   classification  usage_counter        prio  static_prio  normal_prio  \
0               1              0  3069378560        14274            0
1               1              0  3069378560        14274            0
2               1              0  3069378560        14274            0
3               1              0  3069378560        14274            0
4               1              0  3069378560        14274            0

   vm_pgoff  vm_truncate_count  task_size  map_count  hiwater_rss  …  \
0         0              13173          0       6850            0  …
1         0              13173          0       6850            0  …
2         0              13173          0       6850            0  …
3         0              13173          0       6850            0  …
4         0              13173          0       6850            0  …

   exec_vm  reserved_vm  nr_ptes   nvcsw  nivcsw  signal_nvcsw  Debian  Mac  \
0      124          210        0  341974       0             0     0.0  0.0
1      124          210        0  341974       0             0     0.0  0.0
2      124          210        0  341974       0             0     0.0  1.0
3      124          210        0  341974       0             0     0.0  0.0
4      124          210        0  341974       0             0     0.0  1.0

   Ubuntu  Windows
0     0.0      0.0
1     0.0      1.0
2     0.0      0.0
3     1.0      0.0
4     0.0      0.0

[5 rows x 22 columns]
```

[11]:
```
df.describe()
```

[11]:
```
       classification  usage_counter          prio    static_prio  \
count   100000.000000       100000.0  1.000000e+05  100000.000000
mean         0.504130            0.0  3.069706e+09   18183.900070
std          0.505377            0.0  2.963061e+05    4609.792765
```

```
min          0.000000            0.0  3.069190e+09   13988.000000
25%          0.000000            0.0  3.069446e+09   14352.000000
50%          1.000000            0.0  3.069698e+09   16159.000000
75%          1.000000            0.0  3.069957e+09   22182.000000
max          2.000000            0.0  3.070222e+09   31855.000000

       normal_prio  vm_pgoff  vm_truncate_count  task_size     map_count  \
count     100000.0  100000.0      100000.000000   100000.0  100000.00000
mean           0.0       0.0       15312.739510        0.0    8771.13948
std            0.0       0.0        3256.475008        0.0    3785.30516
min            0.0       0.0        9695.000000        0.0    2588.00000
25%            0.0       0.0       12648.000000        0.0    6428.00000
50%            0.0       0.0       15245.000000        0.0    7865.00000
75%            0.0       0.0       17663.000000        0.0   10684.00000
max            0.0       0.0       27157.000000        0.0   28184.00000

       hiwater_rss       total_vm      shared_vm        exec_vm  \
count     100000.0  100000.000000  100000.000000  100000.000000
mean           0.0     266.491120     117.920240     127.678150
std            0.0     311.996779       3.116892      22.277995
min            0.0       4.000000     112.000000      92.000000
25%            0.0      99.000000     114.000000     112.000000
50%            0.0     177.000000     120.000000     127.000000
75%            0.0     327.000000     120.000000     138.000000
max            0.0    2810.000000     120.000000     196.000000

         reserved_vm   nr_ptes          nvcsw         nivcsw  signal_nvcsw
count  100000.000000  100000.0  100000.000000  100000.000000      100000.0
mean      205.324850       0.0  348313.071600      32.991160           0.0
std       112.717875       0.0    9117.720632      52.730176           0.0
min        29.000000       0.0  337688.000000       0.000000           0.0
25%       112.000000       0.0  341974.000000       1.000000           0.0
50%       193.000000       0.0  347244.000000       9.000000           0.0
75%       273.000000       0.0  351667.000000      46.000000           0.0
max       755.000000       0.0  384520.000000     365.000000           0.0
```

[12]:
```
# dropping the Nan as all values of it are zeroes
df_num.drop(['usage_counter', 'normal_prio', 'vm_pgoff', 'task_size',␣
 ↪'hiwater_rss', 'nr_ptes', 'signal_nvcsw'], axis=1, inplace=True)
df_num.head()
```

[12]:
```
   classification        prio  static_prio  vm_truncate_count  map_count  \
0               1  3069378560        14274              13173       6850
1               1  3069378560        14274              13173       6850
2               1  3069378560        14274              13173       6850
3               1  3069378560        14274              13173       6850
4               1  3069378560        14274              13173       6850
```

```
   total_vm  shared_vm  exec_vm  reserved_vm   nvcsw  nivcsw  Debian  Mac  \
0       150        120      124          210  341974       0     0.0  0.0
1       150        120      124          210  341974       0     0.0  0.0
2       150        120      124          210  341974       0     0.0  1.0
3       150        120      124          210  341974       0     0.0  0.0
4       150        120      124          210  341974       0     0.0  1.0

   Ubuntu  Windows
0     0.0      0.0
1     0.0      1.0
2     0.0      0.0
3     1.0      0.0
4     0.0      0.0
```

[34]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer, precision_score, f1_score,␣
 ↪roc_auc_score
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.decomposition import PCA
import pandas as pd

# Assuming 'x' is your feature matrix and 'y' is your target vector

# Scenario 1: Using all features
clf_all_features = LogisticRegression(penalty='l2', solver='lbfgs')
acc_all_features = cross_val_score(clf_all_features, x, y, cv=5,␣
 ↪scoring='accuracy').mean()
pre_all_features = cross_val_score(clf_all_features, x, y, cv=5,␣
 ↪scoring=make_scorer(precision_score, average='macro')).mean()
f1_all_features = cross_val_score(clf_all_features, x, y, cv=5,␣
 ↪scoring=make_scorer(f1_score, average='macro')).mean()
predicted_probabilities = cross_val_predict(clf_all_features, x, y, cv=5,␣
 ↪method='predict_proba')
#Calculate ROC AUC score
auc_all_features = roc_auc_score(y, predicted_probabilities, multi_class='ovr')


# Scenario 2: Using one feature selection method (SelectKBest with k=1)
selector = SelectKBest(score_func=f_classif, k=1)
x_selected = selector.fit_transform(x, y)
clf_selected_features = LogisticRegression(penalty='l2', solver='lbfgs')
acc_selected_features = cross_val_score(clf_selected_features, x_selected, y,␣
 ↪cv=5, scoring='accuracy').mean()
```

```
pre_selected_features = cross_val_score(clf_selected_features, x_selected, y,␣
 ↪cv=5, scoring=make_scorer(precision_score, average='macro')).mean()
f1_selected_features = cross_val_score(clf_selected_features, x_selected, y,␣
 ↪cv=5, scoring=make_scorer(f1_score, average='macro')).mean()
predicted_probabilities = cross_val_predict(clf_selected_features, x_selected,␣
 ↪y, cv=5, method='predict_proba')
#Calculate ROC AUC score
auc_selected_features = roc_auc_score(y, predicted_probabilities,␣
 ↪multi_class='ovr')

# Scenario 3: Using PCA
pca = PCA(n_components=2)
x_pca = pca.fit_transform(x)
clf_pca = LogisticRegression(penalty='l2', solver='lbfgs')
acc_pca = cross_val_score(clf_pca, x_pca, y, cv=5, scoring='accuracy').mean()
pre_pca = cross_val_score(clf_pca, x_pca, y, cv=5,␣
 ↪scoring=make_scorer(precision_score, average='macro')).mean()
f1_pca = cross_val_score(clf_pca, x_pca, y, cv=5, scoring=make_scorer(f1_score,␣
 ↪average='macro')).mean()
predicted_probabilities = cross_val_predict(clf_pca, x_pca, y, cv=5,␣
 ↪method='predict_proba')
#Calculate ROC AUC score
auc_pca = roc_auc_score(y, predicted_probabilities, multi_class='ovr')


# Output
print('Scenario 1 - Using all features:')
print('Accuracy:', acc_all_features)
print('Precision:', pre_all_features)
print('F1 Score:', f1_all_features)
print('AUC Score:', auc_all_features)
print()

print('Scenario 2 - Using one feature selection method:')
print('Accuracy:', acc_selected_features)
print('Precision:', pre_selected_features)
print('F1 Score:', f1_selected_features)
print('AUC Score:', auc_selected_features)
print()

print('Scenario 3 - Using PCA:')
print('Accuracy:', acc_pca)
print('Precision:', pre_pca)
print('F1 Score:', f1_pca)
print('AUC Score:', auc_pca)
```

```
Scenario 1 - Using all features:
```

```
Accuracy: 0.49871
Precision: 0.16623666666666667
F1 Score: 0.22183967092673904
AUC Score: 0.5033163967933176

Scenario 2 - Using one feature selection method:
Accuracy: 0.49858
Precision: 0.16619333333333333
F1 Score: 0.2218010826839746
AUC Score: 0.5537399643294151

Scenario 3 - Using PCA:
Accuracy: 0.59817
Precision: 0.41424896636339625
F1 Score: 0.40091240012926466
AUC Score: 0.6342241478003726
```

[ ]: