ejercicio2-nombre1-apellido1

February 8, 2021

[ ]:
```
""" Cualquier librería adicional que necesiteis durante el ejercicio,␣
 ↪importadlo en esta sección """

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

sns.set_style('darkgrid')
np.set_printoptions(precision=2)
warnings.filterwarnings("ignore")

from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer,␣
 ↪Binarizer, RobustScaler
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, PowerTransformer
from sklearn.impute import SimpleImputer, KNNImputer

from sklearn.feature_selection import SelectKBest, chi2, RFE
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline, Pipeline
from sklearn.decomposition import PCA

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report, f1_score

from sklearn.model_selection import KFold, ShuffleSplit, LeaveOneOut,␣
 ↪StratifiedKFold
```

```
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

1 Ejercicio 2

Este ejercicio pretende poner en práctica la habilidad de limpiar datos y visualizar plots en para crear finalmente modelos en **sklearn**.

El estudiante tendrá que repasar los comandos realizados en clase y lidiar con posibles errores durante el desarrollo.

Para facilitar y agilizar el desarrollo, el estudiante tendrá que rellenar los huecos marcados como '*# codigo-alumno*'. No obstante, si además el estudiante necesita ejecutar código adicional, siempre podrá utilizar cualquier celda adicional.

El estudiante tendrá siempre que introducir una semilla (seed) que generará acorde a su fecha de nacimiento (sin ser intrusivos en edad).

Finalmente, la entrega será un fichero .ipynb cambiando nombre y apellido al fichero.

[ ]:
```
""" El estudiante tendrá que utilizar la semilla proporcionada para todos los␣
 ↪procesos aleatorios """

seed = #dia-nacimiento-estudiante + 13 * mes-nacimiento-estudiante
```

1.0.1 Data cleansing

[ ]:
```
""" Leed el dataframe de sklearn llamado 'fetch_kddcup99' y almacenarlo en una␣
 ↪variable llamada data """

from sklearn.datasets import fetch_kddcup99

data = fetch_kddcup99(as_frame=True)
pd_data = data.frame
pd_data.head()
```

[ ]:
```
""" De entre todas las variables del dataframe, solo trabajaremos en
    este ejercicio con las descritas en key_columns, por lo que tendréis que␣
 ↪quitar el resto """

key_columns = ['duration', 'protocol_type', 'service', 'flag', 'logged_in',␣
 ↪'count', 'srv_count', 'serror_rate', 'dst_host_srv_count',␣
 ↪'dst_host_srv_serror_rate', 'labels']

pd_data = pd_data[key_columns]
pd_data.head()
```

[ ]:
```
""" Comprobad que no haya nulos ni registros duplicados """

# codigo-alumno
```

[ ]:
```
""" Mostrar un barplot para la variable objetivo (labels)"""

target = 'labels'

# codigo-alumno
```

[ ]:
```
""" Dado que hay dos etiquetas que se repiten con mayor frecuencia,
    transformad la variable labels para que tenga un valor booleano que indique
    si es la etiqueta más frecuente o la segunda más frecuente,
    los demás registros los eliminaremos de este estudio """

# codigo-alumno

print(len(pd_data))
pd_data.groupby('labels').size().sort_values(ascending=False)
```

[ ]:
```
""" Volved a mostrar el barplot para la variable objetivo (labels) """

# codigo-alumno
```

[ ]:
```
""" Separaremos las variables categóricas de las numéricas """


num_cols = ['duration', 'count', 'srv_count', 'serror_rate',␣
 ↪'dst_host_srv_count', 'dst_host_srv_serror_rate']
cat_cols = ['protocol_type', 'service', 'flag', 'logged_in']

pd_data[num_cols] = pd_data[num_cols].astype(float)
```

[ ]:
```
""" Mostrad un histograma por cada variable numérica """

# codigo-alumno
```

[ ]:
```
""" Por la forma de los histogramas, podría ser un buen estudio convertir las␣
 ↪variables
    numéricas a variables dummy, es lo que hareis en este apartado y, por␣
 ↪tanto, pasarán
    a ser categóricas todas las variables del dataframe. Esta parte será libre␣
 ↪para el
    estudiante. Deberá tomar la decisión que considere más apropiada para␣
 ↪realizar esta
    binarización """
```

```
# codigo-alumno
```

[ ]:
```
""" Mostrad un barplot por cada variable """

# codigo-alumno
```

[ ]:
```
""" Transformad la variable service en una variable dummy que nos permitan␣
 ↪identificar el servicio
    más frecuente frente al resto """

# codigo-alumno

pd_data.groupby('service').size().sort_values(ascending=False)
```

[ ]:
```
""" Transformad la variable flag en dos variables dummy que nos permitan␣
 ↪identificar 3 tipos de flags,
    los dos más comunes y el resto """

# codigo-alumno

pd_data.groupby('flag').size().sort_values(ascending=False)
```

[ ]:
```
""" Volved a mostrad un barplot por cada variable """

# codigo-alumno
```

[ ]:
```
""" Mostrad un plot de correlaciones entre variables numéricas """

# codigo-alumno
```

1.0.2 Model evaluation

[ ]:
```
""" Realizad una evaluación de, al menos, cinco modelos de machine learning con
    la técnica de validación cruzada más acertada (10 splits). Además, se pide
    incorporar, al menos, una técnica de selección previa de las 1, 2 o 3␣
 ↪features que
    mejores resultados ofrezca (Nota, tendreis que usar OneHotEncoder para las␣
 ↪variables
    que tengan strings) """

X = pd_data.drop(target, axis=1)
y = pd_data[target]

# codigo-alumno
```

[ ]:
```

```

[ ]:

[ ]:

[ ]:

[ ]: