Actividad autocorregible. Conceptos generales de árboles y *random forest* para clasificación (II)

Objetivos

Mediante esta actividad se pretende que pongas en práctica la creación de modelos basados en máquinas de vector de soporte y redes de neuronas. El objetivo es que comprendas de forma práctica con un problema determinado las diferencias que existen a la hora de entrenar los diferentes modelos.

Descripción de la actividad

Para resolver las preguntas aquí planteadas, se deben solucionar los interrogantes del archivo **Act2.ipynb,** adjunto a esta actividad. Una vez solucionado en el cuaderno de Jupyter, debes contestar a las siguientes preguntas.

Preguntas

1. ¿ Qué código permite crear los gráficos de dispersión de todas las variables?

A.

sns.pairplot(data1,hue=’liked’)

B.

plt.figure(figsize=(25,25))

sns.pairplot(data,hue='liked') #Para mapear de diferentes colores

C.

plt.rcParams['figure.figsize'] = [20, 10];

ptl.pairplot(x, bins=number of bins).

plt.show()

D.

ax.pairplot(x, bins=[0,25,50,75,100]).

plt.show()

1. ¿Qué código permite crear la matriz de correlación del *dataset* denominado *data*?

A.

corr=data.corr()

B.

corr(data)

C.

corr(data1)

D. Ninguno de los anteriores códigos lo permite.

1. ¿Qué código me permite reescalar todos los atributos del *dataset* utilizando la función StandardScaler?

A.

Data2=scaler.fit\_transform(data2)

B.

Data=scaler.fit\_transform(data2)

C.

sc = StandardScaler()

data2[['key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', 'duration\_ms', 'time\_signature', 'energy', 'danceability']] = sc.fit\_transform(data2[['key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', 'duration\_ms', 'time\_signature', 'energy', 'danceability']])374

D. Ninguno de los anteriores códigos permite reescalar los datos.

1. ¿Cuál es la variable *target* del conjunto de datos?

A. *Liked.*

B. *Danceability*.

C. *Energy*.

D. *Mode*.

1. ¿Qué código permite crear un modelo de SVM con kernel lineal, valor de C=1 y semilla aleatoria=1234?

A.

modeloSVM = SVC(C = ‘lineal’, kernel = 1, random\_state=1234)

modeloSVM.fit(X\_train, y\_train)

B.

modeloSVM = SVC(C = 1, kernel = 'linear', random\_state=1234)

modeloSVM.fit(X\_test, y\_test)

C.

modeloSVM = SVC(C = 1, kernel = 'lineal', random\_state=123)

modeloSVM.fit(X\_train, y\_train)

D.

modeloSVM = SVC(C = 1, kernel = 'linear', random\_state=1234)

modeloSVM.fit(X\_train, y\_train)

1. El valor de la exactitud del modelo que entrena un SVM lineal y C=1 es:

A. Es menor de 0.1.

B. Está entre 0.3 y 0.5.

C. Está entre 0.6 y 0.8.

D. Es superior a 0.8. = 0.8974

1. ¿Cuál es el código que ayuda a encontrar los mejores parámetros para el modelo SVM con un param\_grid= 'C': np.linspace(0.1, 100, 200), 'kernel': ('linear', 'rbf')?

A.

clas\_rndforest = RandomForestClassifier(n\_estimators=2,n\_jobs=100, random\_state=semilla\_aleatoria)

clas\_rndforest.fit(train\_x,train\_y)

B.

grid = GridSearchCV(

estimator = SVC(),

param\_grid = param\_gride,

scoring = 'accuracy',

n\_jobs = -1,

cv = 5,

verbose = 0,

return\_train\_score = True

)

C.

grid = GridSearchCV(

estimator = SVC(),

param\_grid = param\_grid,

scoring = 'accuracy',

n\_jobs = -1,

cv = 5,

verbose = 0,

return\_train\_score = True)

D.

clas\_rndforest = RandomForestClassifier(n\_estimators=100,n\_jobs=2, random\_state=semilla\_aleatoria)

clas\_rndforest.fit(train\_x,train\_y)

1. El valor de la exactitud del modelo SVM que utiliza los mejores valores hallados en el numeral 3.3:

A. Es menor de 0.1.

B. Está entre 0.6 y 0.8.

C. Está entre 0.5 y 0.95.

D. Es superior a 0.8. – 0.9487

1. Los valores de precisión para las dos clases de la red neuronal entrenada en el apartado 4.1 es de:

A. Para la clase 0, está entre 0.7 y 0.75, y para la clase 1, está entre 0.65 y 0.75.

B. Para las dos clases está entre 0.75 y 0.9.

C. Para la clase 0 es superior a 0.8, y para la clase 1 está entre 0.6 y 0.7.

D. Para las dos clases está entre 0.9 y 1.

1. Los valores de precisión para las dos clases de la red neuronal entrenada en el apartado 4.1 es de:

A. Para la clase 0 está entre 0.7 y 0.75, y para la clase 1 está entre 0.65 y 0.75.

B. Para las dos clases está entre 0.65 y 0.9.

C. Para la clase 1 es superior a 0.8 y para la clase 0 está entre 0.6 y 0.8.

D. Para las dos clases está entre 0.9 y 1.