Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Actividad: Clasificación con máquina de vectores de soporte y redes de neuronas

Introducción

A lo largo del nuestro día se toman variadas decisiones basadas en clasificación, lectores de huellas dactilares, reconocimiento facial, filtros de spam para correo electrónico, etc.

En la inteligencia artificial uno de los métodos más utilizados son las redes neuronales y las Maquinas de Soporte Vectorial (SVM) para realizar problemas tanto de regresión como de clasificación.

El objetivo de esta práctica es conocer mejor cómo funcionan estos algoritmos para realizar un problema de clasificación basado en el precio de los celulares, utilizando un set de datos tomado de la página de kaggle con múltiples datos recolectados.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Desarrollo

Máquina de soporte vectorial (SVM)

La primera parte de la práctica se centra en los SVM, los cuales funcionan correlacionando los datos en un plano de características de grandes dimensiones de forma que los puntos en el plano se puedan categorizar correctamente, se añade un separador el cual es una línea que se encargara de dividir y categorizar los datos correctamente.

Análisis de los datos

Para iniciar la practica primero se deben importar las librerías necesarias y conocer los datos con los cuales se va a trabajar, para esto cargamos el dataset y con la función head () de pandas, observamos algunas de nuestras variables con su respectivo valor.

```
#importamos las librerias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
dfTrain = pd.read_csv("train.csv")
dfTrain.shape
dfTrain.head()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	th
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	

5 rows × 21 columns

Se procede a obtener los posibles valores que puede tomar nuestra variable de interés, en este caso la variable de interés es 'Price_range'.

```
#obtenemos los posibles valores que puede tomar el rango de precio
dfTrain['price_range'].value_counts()
```

```
1    500
2    500
3    500
0    500
Name: price_range, dtype: int64
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Con la realización de una correlación con la variable de interés, se obtienen las variables mas relevantes que influyen en el precio de un celular, en la siguiente imagen se muestran las variables correlacionadas.

```
corrTrain = dfTrain.corr()
print(corrTrain['price_range'].sort_values(ascending=False)[:8],'\n')
```

```
price_range      1.000000
ram              0.917046
battery_power    0.200723
px_width         0.165818
px_height        0.148858
int_memory       0.044435
sc_w             0.038711
pc               0.033599
Name: price_range, dtype: float64
```

Como se observa en la imagen, una de las variables que mas influye en el precio de un teléfono es la ram y las demás variables influyen en menor medida, se procede a realizar un filtrado con estas variables que influyen directamente en el precio, además se realiza un proceso para asegurarnos de que no vengan valores nulos en estas variables.

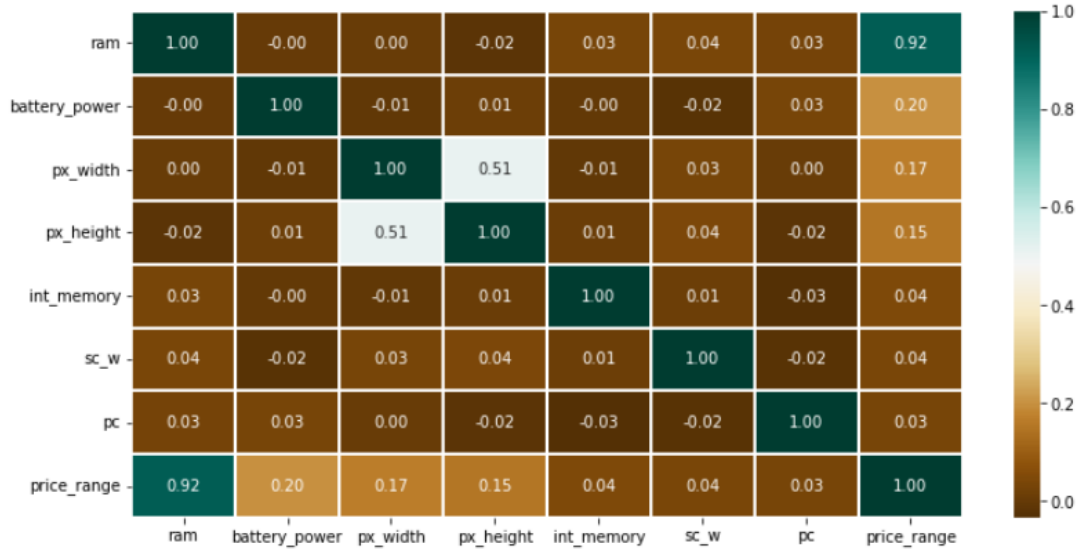
```
dfFilter1 = dfTrain[["ram","battery_power","px_width","px_height","int_memory","sc_w","pc","price_range"]]
dfFilter1.isnull().sum()
```

```
ram              0
battery_power    0
px_width         0
px_height        0
int_memory       0
sc_w             0
pc               0
price_range      0
dtype: int64
```

Una vez realizado el filtrado, se procede nuevamente a realizar la correlación para ver mejor como se relacionan los datos, pero se realiza el proceso de una manera mas grafica para entender mejor nuestras variables respecto al precio de los teléfonos.

```
plt.figure(figsize=(12, 6))
sns.heatmap(dfFilter1.corr(),
            cmap = 'BrBG',
            fmt = '.2f',
            linewidths = 2,
            annot = True)
```

```
<AxesSubplot:>
```



Una vez que se entiende la relación de los datos seleccionados con los de la variable objetivo, se realiza una división entre los datos a analizar y la variable de interés que en este caso es 'price_range', se dividen los datos y se guardan en las variables (X,y), donde la "y" tiene la variable de interés y "X" conserva los demás datos excluyendo la variable "Price_range".

```
y = dfFilter1['price_range']
X = dfFilter1.drop('price_range',axis=1)
```

Se procede a dividir los datos para las pruebas y entrenamiento con la función train_test_split.

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0,stratify=y)
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

La máquina de soporte vectorial funciona mejor con datos escalados, por lo tanto, se realiza un proceso de escalamiento con la librería StandardScaler, se escalarán tanto los sets de test y train.

```
#Escalar los datos
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_array = sc.fit_transform(X_train.values)
X_train = pd.DataFrame(X_train_array, index=X_train.index, columns=X_train.columns)
X_test_array = sc.transform(X_test.values)
X_test = pd.DataFrame(X_test_array, index=X_test.index, columns=X_test.columns)
```

Se procede a realizar el modelo de máquina de soporte vectorial, para esto se utiliza la librería sklearn y se importa el SVC, se especifica el kernel, el cual puede recibir 5 posibles instrucciones *'linear', 'poly', 'rbf', 'sigmoid' y 'precomputed',* para la practica se probo con los 5 y el que dio mejores resultados fue el "linear", también recibe el parámetro gamma para esta practica se utilizo el valor de "scale", otro parámetro importante es el de random_state, el cual se indico con 0 para que no presente aleatoriedad. Una vez realizado el modelo, con la función predict, se realizan las predicciones necesarias en nuestro algoritmo.

```
#Libreria para el proceso de maquina de soporte vectorial
from sklearn.svm import SVC
modelo = SVC(C=2.0,kernel='linear',gamma='scale',random_state=0).fit(X_train,y_train)
# Predicciones test
# ==============================================================================
predicciones = modelo.predict(X_test)
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Rendimiento del modelo MSV

Para evaluar el desempeño de nuestro modelo se realiza el cálculo del accuracy, la cual nos indica que tan cerca están nuestros resultados del valor autentico de nuestro objetivo o meta, también se utiliza la desviación estándar, cual nos indica la dispersión o desviación típica en el conjunto de datos, es decir que tanto se equivocó porcentualmente nuestro modelo al momento de realizar la clasificación.

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = modelo, X = X_train, y = y_train, cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 95.50 %
Standard Deviation: 1.74 %
```

Como se puede observar con este modelo de MSV se obtuvo un accuracy de 95.50 % y una desviación estándar de 1.74%, lo cual es bastante bueno, pues se puede decir que el modelo esta lo suficientemente capacitado para clasificar correctamente los datos con un margen de error muy poco.

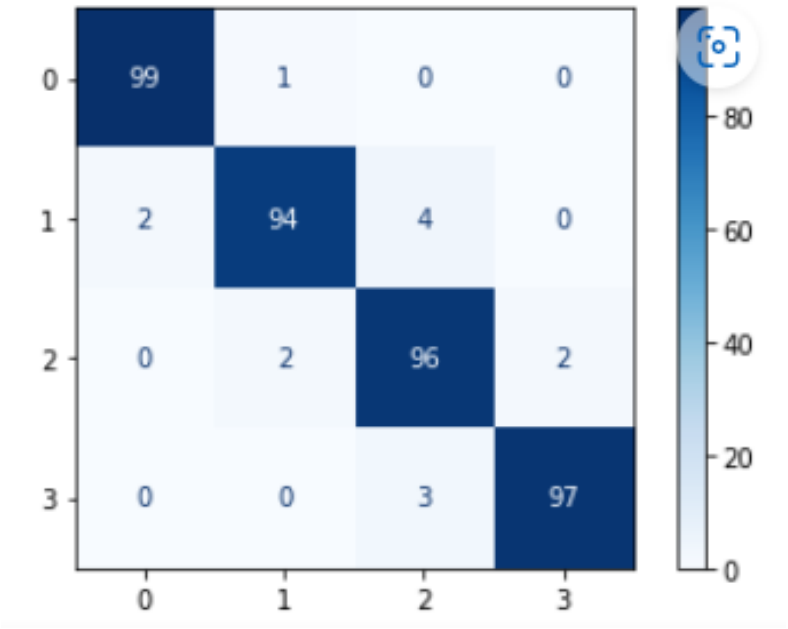
Otra forma de evaluar el modelo es utilizar una matriz de confusión, la cual es ampliamente utilizada en el campo de la IA, ya que es una herramienta que sirve para evaluar el desempeño de un algoritmo clasificatoria que se emplee en aprendizaje supervisado.

```
#sacamos la matriz de confusión
from sklearn.metrics import ConfusionMatrixDisplay,classification_report
np.set_printoptions(precision=2)

title_options = [
    ("Matriz de confusión sin normalizar", None),
    ("Matriz de confusión normalizada", "true"),
]
for title, normalize in title_options:
    display = ConfusionMatrixDisplay.from_estimator(
        modelo,
        X_test,
        y_test,
        display_labels=dfFilter1[['price_range']],
        cmap=plt.cm.Blues,
        normalize=normalize,
    )
    display.ax_.set_title(title)

    print(title)
    print(display.confusion_matrix)

plot.show()
```



En la matriz se puede ver como clasifica los rangos de precios 0,1,2 o 3, se observan pequeños errores en el que por ejemplo se equivoca 2 veces al querer poner el precio en el rango de 0 y lo asigna en el valor de 1, pero estos son errores menores y se muestra como presenta un gran desempeño el modelo.

Por último, para ver mejor el desempeño se realiza un reporte de clasificación, en el cual se muestran los datos mas relevantes como la precisión, el re-call e incluso el f-Score.

```
#reporte de clasificasion
print(classification_report(y_test,predicciones))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       100
           1       0.97      0.94      0.95       100
           2       0.93      0.96      0.95       100
           3       0.98      0.97      0.97       100

    accuracy                           0.96       400
   macro avg       0.97      0.96      0.97       400
weighted avg       0.97      0.96      0.97       400
```

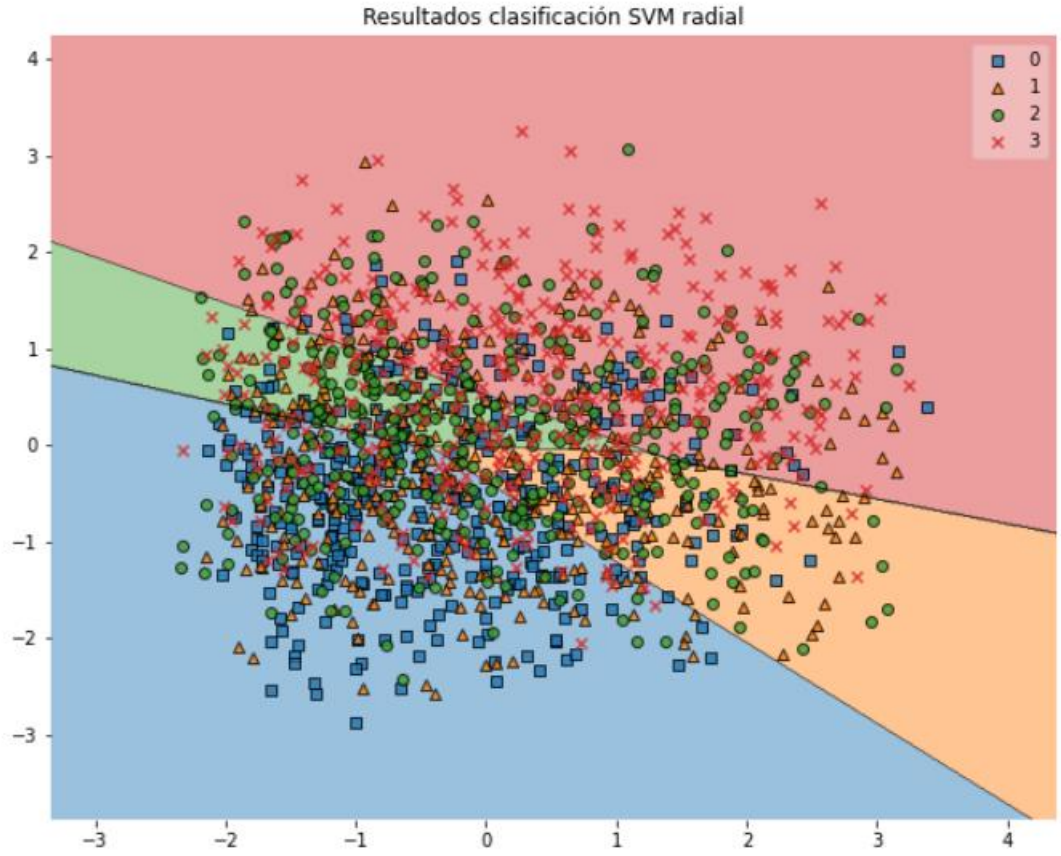
Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Resultados del modelo MSV

Para ver el proceso de clasificación que utiliza el modelo se realiza una representación gráfica utilizando plot_decision_regions() de mlxtend.

```
from mlxtend.plotting import plot_decision_regions
from sklearn.decomposition import PCA
# Representación gráfica utilizando plot_decision_regions() de mlxtend
# ==============================================================================
pca = PCA(n_components = 2)
X_train2 = pca.fit_transform(X_train)
modelo.fit(X_train2, y_train)

fig, ax = plt.subplots(figsize=(10,8))
plot_decision_regions(
    X = X_train2,
    y = y_train.values.flatten(),
    clf = modelo,
    ax = ax
)
ax.set_title("Resultados clasificación SVM radial");
```



En este grafico se observa como clasifica los valores de rango de precio 0, 1, 2 o 3 en su respectiva zona, al ser muchos datos no se alcanza a apreciar tan bien como realiza la clasificación.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Redes neuronales

La segunda parte de la práctica se centra en las redes neuronales de clasificación, los cuales funcionan con un nivel de neuronas interconectadas entre si, tal y como funcionan nuestro cerebro, son un conjunto de nodos que se configuran de tal modo que aportan una solución al problema de clasificación.

Análisis de los datos

Para comenzar el ejercicio, primero importa las librerías necesarias y conoce los datos con los que quieres trabajar, para esto cargamos un dataset y usando la función pandas head() observamos algunas variables y sus valores correspondientes.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,roc_curve,auc,classification_report
```

```
dfTrain = pd.read_csv("train.csv")
dfTrain.head()
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	thi
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	

5 rows × 21 columns

Seguimos obteniendo los posibles valores que puede tomar nuestra variable de interés, en este caso la variable de interés es 'Price_range'.

```
#obtenemos los posibles valores que puede tomar el rango de precio
dfTrain['price_range'].value_counts()
```

```
1    500
2    500
3    500
0    500
Name: price_range, dtype: int64
```

Se observan las demás variables con su respectiva información.

```
#observamos la información de las demas variables
dfTrain.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
 10  pc             2000 non-null   int64  
 11  px_height      2000 non-null   int64  
 12  px_width       2000 non-null   int64  
 13  ram            2000 non-null   int64  
 14  sc_h           2000 non-null   int64  
 15  sc_w           2000 non-null   int64  
 16  talk_time      2000 non-null   int64  
 17  three_g        2000 non-null   int64  
 18  touch_screen   2000 non-null   int64  
 19  wifi           2000 non-null   int64  
 20  price_range    2000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

Correlacionando con la variable de interés se pueden obtener las variables más relevantes que afectan al precio de un teléfono móvil. La siguiente figura muestra las variables relevantes.

```
corrTrain = dfTrain.corr()
print(corrTrain['price_range'].sort_values(ascending=False)[:8],'\n')
```

```
price_range      1.000000
ram              0.917046
battery_power    0.200723
px_width         0.165818
px_height        0.148858
int_memory       0.044435
sc_w             0.038711
pc               0.033599
Name: price_range, dtype: float64
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Como se observa en la imagen, una de las variables que mas influye en el precio de un teléfono es la ram y las demás variables influyen en menor medida, se procede a realizar un filtrado con estas variables que influyen directamente en el precio, además se realiza un proceso para asegurarnos de que no vengan valores nulos en estas variables.

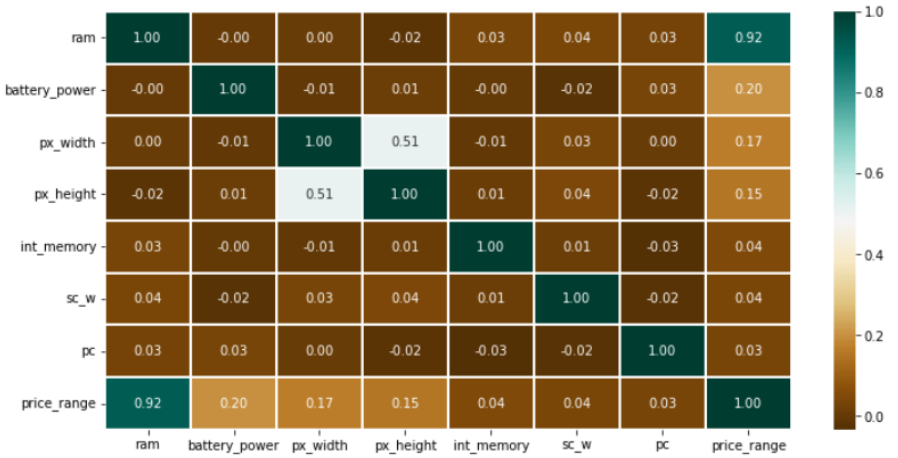
```
dfFilter1 = dfTrain[["ram","battery_power","px_width","px_height","int_memory","sc_w","pc","price_range"]]
dfFilter1.isnull().sum()
```

```
ram             0
battery_power   0
px_width        0
px_height       0
int_memory      0
sc_w            0
pc              0
price_range     0
dtype: int64
```

Una vez hecho el filtrado, volvemos a hacer la correlación para tener una mejor idea de la relevancia de los datos, pero de forma más gráfica para tener una mejor idea de las variables de precio del teléfono.

```
plt.figure(figsize=(12, 6))
sns.heatmap(dfFilter1.corr(),
            cmap = 'BrBG',
            fmt = '.2f',
            linewidths = 2,
            annot = True)
```

```
<AxesSubplot:>
```



Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Una vez que se entiende la relación de los datos seleccionados con los de la variable objetivo, se realiza una división entre los datos a analizar y la variable de interés que en este caso es 'price_range', se dividen los datos y se guardan en las variables (X,y), donde la "y" tiene la variable de interés y "X" conserva los demás datos excluyendo la variable "Price_range".

```
y = dfFilter1['price_range']
X = dfFilter1.drop('price_range',axis=1)
```

Se procede a realizar un escalamiento de los datos para que el algoritmo funcione mejor.

```
sc = StandardScaler()
Xf = sc.fit_transform(X)
Xf
```

```
array([[ 0.39, -0.9 , -1.15, ..., -1.38,  0.28, -1.31],
       [ 0.47, -0.5 ,  1.7 , ...,  1.16, -0.64, -0.65],
       [ 0.44, -1.54,  1.07, ...,  0.49, -0.86, -0.65],
       ...,
       [ 0.86,  1.53,  0.88, ...,  0.22, -1.09, -1.14],
       [-1.16,  0.62, -1.35, ...,  0.77,  0.97, -0.81],
       [ 1.66, -1.66, -1.15, ...,  0.71, -0.41,  1.  ]])
```

Con las variables de salida, se realiza un procedimiento para cambiar los valores categóricos en binarios, ya que las redes neuronales trabajan mejor con valores de 0 y 1.

```
yf = pd.get_dummies(y,columns = ['price_range'])
```

```
yf
```

	0	1	2	3
0	0	1	0	0
1	0	0	1	0
2	0	0	1	0
3	0	0	1	0
4	0	1	0	0
...	...	...	...	...
1995	1	0	0	0
1996	0	0	1	0
1997	0	0	0	1
1998	1	0	0	0
1999	0	0	0	1

2000 rows × 4 columns

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Una vez realizado el proceso de tratamiento de datos, se realiza la división para el set de entrenamiento y el set de test.

```
from sklearn.model_selection import train_test_split, GridSearchCV
x_train, x_test, y_train, y_test = train_test_split(Xf, yf, test_size=0.3, random_state=0)
```

Para realizar la red neuronal, se instala tensorFlow en caso de no estar instalado, mediante la siguiente instrucción se puede realizar la instalación en jupyter notebook.

```
pip install tensorflow
```

```
Collecting tensorflow
  Downloading tensorflow-2.11.0-cp39-cp39-win_amd64.whl (1.9 kB)
Collecting tensorflow-intel==2.11.0
  Downloading tensorflow_intel-2.11.0-cp39-cp39-win_amd64.whl (266.3 MB)
Collecting tensorflow-io-gcs-filesystem>=0.23.1
  Downloading tensorflow_io_gcs_filesystem-0.31.0-cp39-cp39-win_amd64.whl (1.5 MB)
Collecting astunparse>=1.6.0
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting libclang>=13.0.0
  Downloading libclang-15.0.6.1-py2.py3-none-win_amd64.whl (23.2 MB)
Requirement already satisfied: h5py>=2.9.0 in c:\users\jcarl\anaconda3\lib\site-packages (from tensorflow-intel==2.11.0->tensor
flow) (3.6.0)
Collecting tensorflow-estimator<2.12,>=2.11.0
  Downloading tensorflow_estimator-2.11.0-py2.py3-none-any.whl (439 kB)
Collecting opt-einsum>=2.3.2
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\jcarl\anaconda3\lib\site-packages (from tensorflow-intel==2.11.0
->tensorflow) (1.42.0)
Collecting keras<2.12,>=2.11.0
  Downloading keras-2.11.0-py2.py3-none-any.whl (1.7 MB)
Collecting google-pasta>=0.1.1
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\jcarl\anaconda3\lib\site-packages (from tensorflow-intel==2.11.0->tens
orflow) (1.12.1)
Collecting flatbuffers>=2.0
```

Una vez instalada la librería tensorFlow se procede a importar la librería y a crear la red neuronal con la siguiente instrucción.

```
import tensorflow as tf
from keras.models import Sequential

#creación de la red neuronal
model = Sequential()
model.add(tf.keras.layers.Dense(32, input_dim=7, activation='relu'))
model.add(tf.keras.layers.Dense(16, activation='relu'))
model.add(tf.keras.layers.Dense(20, activation='sigmoid'))
model.add(tf.keras.layers.Dense(4, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Una vez creada la red neuronal se realiza un escalamiento de los datos para que sean valores mejor procesados por nuestra red, mediante la librería StandardScaler y una vez escalado nuestros datos se realiza un entrenamiento con los datos ya escalados.

```
from sklearn.preprocessing import StandardScaler
std_clf = StandardScaler()
x_train_new = std_clf.fit_transform(x_train)
```

```
model.fit(x_train_new,y_train, epochs=300,verbose= False)
```

```
<keras.callbacks.History at 0x24fd3a00dc0>
```

Rendimiento del modelo Red neuronal

Para evaluar el modelo creado anteriormente a partir de una red neuronal, se realiza un calculo de accuracy tal y como se muestra en la siguiente imagen.

```
#evaluacion del desempeño de la red neuronal
rend = model.evaluate(x_train,y_train,verbose=False)
print("Training Accuracy: %.2f%%\n" % (rend[1]*100))
rend2 = model.evaluate(x_test,y_test,verbose=False)
print("Testing Accuracy: %.2f%%\n" % (rend2[1]*100))
```

```
Training Accuracy: 99.93%

Testing Accuracy: 95.00%
```

Como se observa en la imagen, el modelo esta bien entrenado tanto con los datos de training como los de testing, por lo cual se puede decir que realiza las predicciones en su mayoría bien y en este caso no presenta sobre entrenamiento por lo cual no hay necesidad de realizar algún ajuste extra.

Una vez revisada el accuracy se realizan las predicciones con los datos de test.

```
#Realizando prediccinoes con los datos de test
y_testpred = model.predict(x_test)
y_testpred = np.round(y_testpred)
```

```
19/19 [==============================] - 0s 818us/step
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Para observar mejor el desempeño del modelo, se realiza una matriz de confusión y para realizarla primero se predicciones de salida con los valores máximos.

```
y_finpredict = (np.argmax(y_testpred,axis=1)+1) #prediccion de la salida con valores maximo

y_testfin=[]
for i in range(len(y_test)):
    if (y_test.iloc[i,0]==1):
        y_testfin.append(1)
    if (y_test.iloc[i,1]==1):
        y_testfin.append(2)
    if (y_test.iloc[i,2]==1):
        y_testfin.append(3)
    if (y_test.iloc[i,3]==1):
        y_testfin.append(4)
print(y_testfin)
```

```
[4, 1, 3, 3, 3, 1, 1, 4, 4, 2, 2, 4, 1, 3, 4, 1, 4, 3, 3, 2, 1, 1, 4, 2, 3, 3, 4, 2, 4, 2, 2, 1, 3, 1, 2, 4, 1, 1, 4, 4, 4, 2,
4, 4, 2, 4, 1, 2, 4, 2, 2, 4, 1, 4, 1, 4, 3, 3, 1, 4, 4, 2, 4, 3, 2, 3, 4, 3, 3, 3, 4, 3, 2, 1, 2, 4, 3, 3, 2, 3, 4, 4, 4, 1,
1, 1, 3, 2, 3, 4, 2, 3, 3, 2, 1, 4, 4, 4, 1, 4, 2, 2, 3, 2, 4, 3, 3, 4, 3, 4, 4, 1, 1, 2, 4, 4, 1, 1, 2, 1, 1, 4, 3, 3, 2, 2,
2, 2, 1, 3, 2, 4, 3, 4, 4, 4, 4, 3, 1, 2, 2, 3, 2, 4, 2, 4, 1, 1, 3, 1, 2, 2, 2, 2, 4, 1, 1, 4, 2, 4, 3, 2, 4, 2, 3, 4, 4, 3,
2, 1, 4, 2, 3, 4, 4, 1, 3, 3, 4, 1, 3, 2, 1, 2, 3, 2, 3, 1, 4, 4, 2, 2, 1, 3, 4, 1, 2, 3, 3, 1, 4, 4, 4, 2, 3, 4, 4, 4, 1, 1,
1, 3, 4, 4, 1, 1, 2, 4, 2, 4, 4, 4, 1, 1, 3, 4, 4, 2, 1, 3, 1, 1, 1, 4, 3, 1, 3, 3, 2, 2, 1, 3, 4, 4, 1, 1, 2, 4, 4, 3, 4, 1,
4, 2, 2, 1, 3, 4, 4, 3, 1, 1, 2, 3, 4, 3, 3, 4, 3, 2, 1, 4, 4, 3, 2, 4, 3, 3, 3, 2, 1, 3, 3, 2, 1, 1, 3, 3, 3, 3, 1, 2, 4, 1,
2, 3, 4, 1, 3, 1, 2, 2, 4, 1, 1, 3, 4, 2, 3, 1, 3, 1, 4, 1, 4, 4, 3, 4, 2, 3, 3, 2, 2, 2, 1, 2, 1, 4, 2, 1, 4, 1, 1, 2, 4, 1,
4, 2, 3, 1, 2, 4, 1, 3, 3, 2, 3, 2, 2, 1, 3, 1, 1, 4, 2, 3, 4, 3, 3, 1, 4, 3, 3, 2, 4, 3, 4, 4, 4, 1, 3, 1, 4, 1, 2, 2, 3, 3,
2, 4, 2, 3, 1, 2, 3, 4, 1, 1, 2, 4, 1, 4, 1, 3, 3, 2, 2, 1, 3, 2, 1, 2, 4, 1, 4, 4, 1, 3, 2, 4, 2, 2, 4, 3, 1, 4, 3, 3, 1, 1,
4, 1, 2, 2, 2, 4, 3, 4, 3, 1, 4, 1, 1, 2, 4, 1, 1, 4, 3, 3, 3, 4, 1, 1, 2, 3, 2, 3, 1, 4, 4, 1, 4, 4, 1, 3, 3, 2, 1, 3, 3, 2,
4, 3, 3, 1, 3, 1, 4, 4, 3, 2, 1, 4, 2, 3, 1, 1, 2, 4, 1, 4, 1, 1, 3, 3, 1, 2, 4, 1, 4, 3, 2, 3, 1, 4, 2, 3, 4, 3, 3, 3, 4, 3,
4, 4, 1, 1, 3, 1, 4, 2, 2, 2, 1, 4, 3, 1, 3, 1, 4, 2, 2, 1, 3, 2, 1, 4, 2, 4, 1, 1, 3, 1, 4, 2, 1, 1, 3, 4, 1, 1, 1, 2, 3, 4,
2, 4, 3, 3, 1, 3, 2, 3, 4, 2, 3, 2, 3, 4, 3, 3, 2, 4, 1, 4, 1, 1, 2, 2, 3, 1, 2, 1, 2, 4, 4, 2, 4, 1, 3, 1, 1, 2, 4, 2, 3, 2,
4, 3, 2, 2, 3, 3, 1, 2, 4, 1, 3, 3]
```

Una vez se obtienen estos datos se crea la matriz de confusión y se grafica.

```
#realizando la matriz de confusion
cm = confusion_matrix(y_finpredict,y_testfin)
```

```
print(cm)
```

```
[[150   5   1   0]
 [  1 127   7   0]
 [  0   3 137   7]
 [  0   0   6 156]]
```

Como se observa el modelo, clasifica correctamente los datos, salvo en unas ocasiones donde se equivoca y los coloca en otra fila o columna que no debería ser, esto es normal ya que nuestro modelo no presenta un 100% de accuracy por lo cual quiere decir que no es perfecto, pero viendo la matriz y observando los datos se llega a la conclusión de que realiza un trabajo bien hecho a la hora de clasificar los datos.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

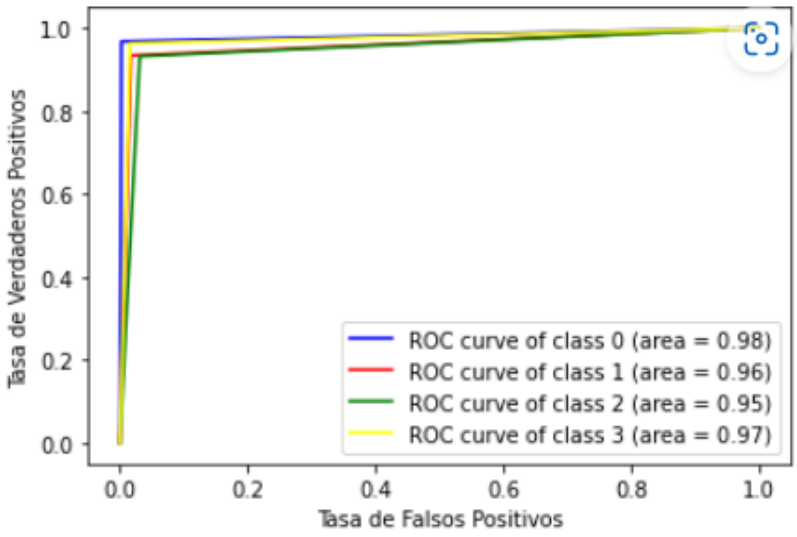
Otra forma de evaluar el modelo, es mediante la curva ROC, la cual es un procedimiento que se usa ampliamente en la ciencia de datos e IA para evaluar modelos de clasificación, el sitio oficial nos da la siguiente definición para entender mejor de que van estas curvas.

"El Análisis ROC (Receiver operating characteristic o característica operativa del receptor) es una forma útil de evaluar la precisión de las predicciones de modelo al trazar la sensibilidad frente a la especificidad de una prueba de clasificación (ya que el umbral varía en todo un rango de resultados de pruebas de diagnostico). El Análisis ROC admite la inferencia en relación con una sola AUC, curvas de precisión/exhaustividad (PR), y proporciona opciones para comparar dos curvas ROC que se generan de grupos independientes o sujetos emparejados."

```
#curva rogue
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(4):
    print(i)
    fpr[i],tpr[i],_= roc_curve(y_testpred[:,i],y_test.iloc[:,i])
    roc_auc[i] = auc(fpr[i],tpr[i])
colors = ['blue','red','green','yellow']
for i,color in zip(range(4),colors):
    plt.plot(fpr[i],tpr[i],color = color,
    label = 'ROC curve of class {0} (area = {1:0.2f})'
    ''.format(i, roc_auc[i]))

plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.legend(loc = 'lower right')
plt.show()
```

```
0
1
2
3
```



Con estas curvas Roc se puede observar que nuestro modelo tiene pocos errores de falsos positivos.

Por último, para finalizar el análisis de desempeño del modelo, se realiza un reporte completo de rendimiento.

```
#reporte de clasificasion
print(classification_report(y_test,y_testpred))
```

	precision	recall	f1-score	support
0	0.97	0.99	0.98	151
1	0.93	0.94	0.94	135
2	0.93	0.91	0.92	151
3	0.96	0.96	0.96	163
micro avg	0.95	0.95	0.95	600
macro avg	0.95	0.95	0.95	600
weighted avg	0.95	0.95	0.95	600
samples avg	0.95	0.95	0.95	600

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Conclusiones

Las técnicas observadas en esta práctica, tanto redes neuronales como maquinas de soporte vectorial, me parece de una gran utilidad para temas de clasificación, ambos modelos presentaron resultados excelentes y su desempeño fue bastante bueno para este set de datos analizado. Sin embargo, en gustos personales preferiría las redes neuronales ya que permiten una mejor personalización y ajuste, por lo cual creo que se pueden adaptar mejor a otros tipos de set de datos y problemas, sin embargo, para el caso de las maquinas de soporte vectorial me parecieron bastante fáciles de implementar, además de dar resultados bastantes óptimos.

Sin duda ambas estrategias funcionan muy bien para realizar problemas de clasificación, en un futuro se pondrán a prueba con algún problema de regresión para ver que tal se comportan ambos modelos, ya que según leí en varias paginas estas dos estrategias funcionan tanto para regresión como para clasificación.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Romero perez	06/03/2023
	Nombre: Juan carlos	

Bibliografía

TensorFlow (S.F). Clasificacion Basica: Predecir una imagen de moda Recuperado el 27 de 02 de 2022, de https://www.tensorflow.org/tutorials/keras/classification?hl=es-419

Máquinas de Vector Soporte (SVM) con Python by Joaquín Amat Rodrigo, available under a Attribution 4.0 International (CC BY 4.0) at https://www.cienciadedatos.net/documentos/py24-svm-python.html

IBM (2021). *IBM*-Análisis ROC. Recuperado el 01 de 03 de 2022, de https://www.ibm.com/docs/es/spss-statistics/beta?topic=features-roc-analysis.