Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	09/05/2022
	Nombre: Aldo Alberto	

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

Objetivos

Se analizará el archivo DataSet mobile-price-classification#train.csv para aplicar Redes Neuronales y Máquinas de vectores con la finalidad de predecir la columna "price_range" , es decir, el precio de los distintos móviles que se listan en el archivo.

Librerías a usar

A continuación, se listan las librerías a usar en el trabajo.

```
In [58]: import matplotlib.pyplot as plt
         import seaborn as sns
         import numpy as np
         import pandas as pd
         import tensorflow as tf
         from sklearn.svm import SVC
         from sklearn.model_selection import train_test_split
         from sklearn.model_selection import GridSearchCV
         from sklearn.metrics import accuracy_score
         from funpymodeling.exploratory import freq_tbl , profiling_num
         from sklearn.model_selection import train_test_split as tts
         from sklearn import metrics
         from sklearn.neural_network import MLPClassifier as MLP
         from sklearn.metrics import recall_score
         from sklearn.metrics import classification_report
         from sklearn.metrics import f1_score
         from sklearn.metrics import precision_score
```

Análisis Descriptivo de los datos

Empezaremos el análisis cargando el archivo "train.csv", asimismo, daremos un rápido vistazo al tipo de datos que contiene cada columna de la Base de Datos.

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	09/05/2022
	Nombre: Aldo Alberto	

```
In [59]: data = pd.read_csv("train.csv")
         data.info()
         #Columnas categoricas estan hechas con 0 y 1
         #Blue , Dual_Sim , Four_g , three_g , touch_screen , wifi
```

```
<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
```

Notaremos inmediatamente que todas las columnas son tipo Float64 o Int64, esto significa que son datos numéricos. Por otra parte, al abrir la base de datos nos percatamos de que en realidad existen 6 columnas, y no obstante que sean números, en realidad son datos categóricos, toda vez que contienen un 0 (false) y 1 (true). Para poder entender y hacer el análisis de la información se decidió cambiar algunos datos para facilitar su lectura. **Este cambio de datos solamente se realizará para el análisis, por lo que al momento de efectuar las operaciones pertinentes se mantendrá el archivo original sin cambio alguno.**

```
data["blue"] = data["blue"].replace( 0 , "No")
data["blue"] = data["blue"].replace( 1 , "Si")
data["dual_sim"] = data["dual_sim"].replace( 0 , "No")
data["dual_sim"] = data["dual_sim"].replace( 1 , "Si")
data["four_g"] = data["four_g"].replace( 0 , "No")
data["four_g"] = data["four_g"].replace( 1 , "Si")
data["three_g"] = data["three_g"].replace( 0 , "No")
data["three_g"] = data["three_g"].replace( 1 , "Si")
data["touch_screen"] = data["touch_screen"].replace( 0 , "No")
data["touch_screen"] = data["touch_screen"].replace( 1 , "Si")
data["wifi"] = data["wifi"].replace( 0 , "No")
data["wifi"] = data["wifi"].replace( 1 , "Si")
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	09/05/2022
	Nombre: Aldo Alberto	

A continuación, se muestran aspectos estadísticos de los datos numéricos.

```
data_numericos = profiling_num(data)
data_numericos
```

	variable	mean	std_dev	variation_coef	p_0.01	p_0.05	p_0.25	p_0.5	p_0.75	p_0.95	p_0.99
0	battery_power	1238.51850	439.418206	0.354793	510.00	570.95	851.75	1226.0	1615.25	1930.15	1987.00
1	clock_speed	1.52225	0.816004	0.536051	0.50	0.50	0.70	1.5	2.20	2.80	3.00
2	fc	4.30950	4.341444	1.007412	0.00	0.00	1.00	3.0	7.00	13.00	16.00
3	int_memory	32.04650	18.145715	0.566231	2.00	5.00	16.00	32.0	48.00	61.00	64.00
4	m_dep	0.50175	0.288416	0.574819	0.10	0.10	0.20	0.5	0.80	1.00	1.00
5	mobile_wt	140.24900	35.399655	0.252406	80.00	86.00	109.00	141.0	170.00	196.00	199.00
6	n_cores	4.52050	2.287837	0.506103	1.00	1.00	3.00	4.0	7.00	8.00	8.00
7	pc	9.91650	6.064315	0.611538	0.00	0.00	5.00	10.0	15.00	20.00	20.00
8	px_height	645.10800	443.780811	0.687917	15.00	70.95	282.75	564.0	947.25	1485.05	1791.01
9	px_width	1251.51550	432.199447	0.345341	512.99	579.85	874.75	1247.0	1633.00	1929.05	1987.00
10	ram	2124.21300	1084.732044	0.510651	296.99	445.00	1207.50	2146.5	3064.50	3826.35	3958.01
11	sc_h	12.30650	4.213245	0.342359	5.00	6.00	9.00	12.0	16.00	19.00	19.00
12	sc_w	5.76700	4.356398	0.755401	0.00	0.00	2.00	5.0	9.00	14.00	17.00
13	talk_time	11.01100	5.463955	0.496227	2.00	3.00	6.00	11.0	16.00	20.00	20.00
14	price_range	1.50000	1.118314	0.745542	0.00	0.00	0.75	1.5	2.25	3.00	3.00

En seguida, se muestran aspectos estadísticos de los datos categóricos.

```
data_categoricos= freq_tbl(data)
```

```
   blue  frequency  percentage  cumulative_perc
0    No       1010       0.505            0.505
1    Si        990       0.495            1.000

-----------------------------------------------------------------

   dual_sim  frequency  percentage  cumulative_perc
0        Si       1019      0.5095           0.5095
1        No        981      0.4905           1.0000

-----------------------------------------------------------------

   four_g  frequency  percentage  cumulative_perc
0      Si       1043      0.5215           0.5215
1      No        957      0.4785           1.0000

-----------------------------------------------------------------

   three_g  frequency  percentage  cumulative_perc
0       Si       1523      0.7615           0.7615
1       No        477      0.2385           1.0000

-----------------------------------------------------------------

   touch_screen  frequency  percentage  cumulative_perc
0            Si       1006       0.503            0.503
1            No        994       0.497            1.000

-----------------------------------------------------------------

   wifi  frequency  percentage  cumulative_perc
0    Si       1014       0.507            0.507
1    No        986       0.493            1.000

-----------------------------------------------------------------
```

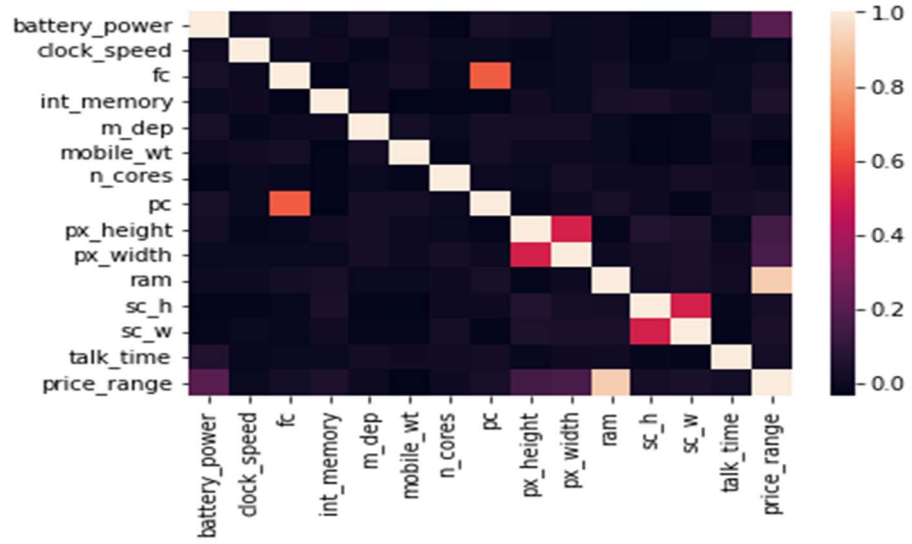
Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	09/05/2022
	Nombre: Aldo Alberto	

Matriz de Correlaciones

Una matriz de correlación es una tabla que contiene coeficientes de correlación entre variables. Cada celda de la tabla representa la correlación entre dos variables, el valor se encuentra entre -1 y 1. Se utiliza una matriz de correlación para resumir los datos, como diagnóstico para análisis avanzados y como entrada para un análisis más avanzado.

```
plt.figure()
sns.heatmap(data.corr())
```

```
<AxesSubplot:>
```



En la Matriz de Correlaciones podemos observar como las columnas sc_h (screen height) y sc_w (screen weight), tienen una mayor relevancia en los datos, lo cual en un análisis en la vida real, tanto **el ancho como el alto del móvil tienen una gran importancia para el precio de este.**

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	09/05/2022
	Nombre: Aldo Alberto	

Redes Neuronales

A continuación se aplican las técnicas de Redes Neuronales para poder predecir price_range . Como se mencionó al inicio del presente trabajo, se volverá a cargar la Base de Datos para poder aplicar las operaciones pertinentes.

```
df = pd.read_csv("train.csv")
```

Posteriormente, ya con la Base de Datos cargada en el DataFrame , se empieza a crear el modelo y los datos de entrenamiento. Empezamos con los datos de entrenamiento para X y Y:

```
#creacion del modelo y datos de entrenamiento
train, test = tts(df, test_size = 0.3)
trainX = train[["battery_power",
                "clock_speed",
                "fc","int_memory",
                "m_dep","mobile_wt",
                "n_cores",
                "pc"
                ,"px_height"
                ,"px_width"
                ,"ram"
                ,"sc_h"
                ,"sc_w"
                ,"talk_time"
                ,"price_range"
                ,"blue"
                ,"dual_sim"
                ,"four_g"
                ,"three_g"
                ,"touch_screen"
                ,"wifi"
                ]]
trainY=train.price_range
```

Continuamos con los datos de práctica o test para X y Y:

```
trainY=train.price_range
testX = train[["battery_power"
               ,"clock_speed"
               ,"fc"
               ,"int_memory"
               ,"m_dep"
               ,"mobile_wt"
               ,"n_cores"
               ,"pc"
               ,"px_height"
               ,"px_width"
               ,"ram"
               ,"sc_h"
               ,"sc_w"
               ,"talk_time"
               ,"price_range"
               ,"blue"
               ,"dual_sim"
               ,"four_g"
               ,"three_g"
               ,"touch_screen"
               ,"wifi"
               ]]
testY=train.price_range
```

Creamos el modelo y entrenamos a la Red Neuronal para su predicción.

```
mlp = MLP(solver='lbfgs', random_state=1, \
    hidden_layer_sizes=[10], max_iter=1000)
mlp.fit(trainX, trainY)
predicciones = mlp.predict(testX)
```

En seguida, realizaremos un análisis del algoritmo de clasificación creado (Red Neuronal) para medir su rendimiento. **Se entiende que es un problema de clasificación multiclase, ya que se tienen varias opciones para definir el precio del móvil: 0,1,2 y 3**

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	09/05/2022
	Nombre: Aldo Alberto	

Evaluación del modelo de Red Neuronal

A continuación, se muestran varias métricas para la evaluación del algoritmo, usaremos los siguientes:

Accuracy. - Representa la proporción del número de predicciones correctas entre el número total de predicciones.

Precisión. - Compromisos hechos en la clasificación, indican lo interesantes y relevantes que son los resultados de un modelo. Indica la proporción de ejemplos que son **verdaderamente positivos** (PPV)

Recall. - Indica que tan completos son los resultados

F1 score. - Es una métrica que combina precisión y recall utilizando la media.

```
# Classification report
print(classification_report(y_true= testY,y_pred=predicciones))
```

```
              precision    recall  f1-score   support

           0       0.80      0.80      0.80       348
           1       0.63      0.48      0.54       359
           2       0.38      0.12      0.18       346
           3       0.51      0.99      0.68       347

    accuracy                           0.60      1400
   macro avg       0.58      0.60      0.55      1400
weighted avg       0.58      0.60      0.55      1400
```

Se puede observar que se tiene una precisión baja, siendo la más alta de .80 y de ahí desciende considerablemente a .38, teniendo un promedio de .58. Es decir, cuando el modelo predice la clase positiva ¿Cuántas veces está en lo cierto? nos indica que **los verdaderos positivos del modelo son bajos**.

Una de las métricas más importantes para poder leer el rendimiento de un modelo de clasificación es F1, ya que como se había explicado anteriormente, es la media entre la precisión y el recall, simplificando el rendimiento a una única métrica. Al igual que Precisión, podemos observar la tendencia a la baja mientras más se hace un Score.

Matriz de confusión del modelo Red Neuronal

Es una tabla que organiza las predicciones en función de los valores reales de los datos. La categoría de interés se conoce como **clase positiva**, mientras las otras son **clase negativa**. Estas instancias caen en la diagonal de la matriz. Los valores fuera de la diagonal indican las instancias clasificadas incorrectamente, la lectura de esta matriz se basa en cuantas instancias caen dentro y fuera de la diagonal de **True Positive.** A continuación, se muestran 2 matrices de confusión, la primera es la que el programa nos muestra al momento de correr el código, y la segunda muestra la lectura que se hace como **True Positive** de las instancias mostradas.

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_true= testY,y_pred=predicciones)
```

```
array([[280,  62,   5,   1],
       [ 68, 171,  59,  61],
       [  3,  40,  41, 262],
       [  0,   0,   4, 343]], dtype=int64)
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_true= testY,y_pred=predicciones)
```

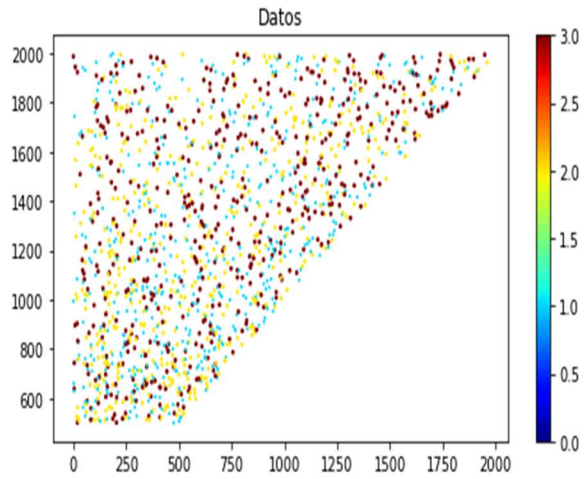
```
array([[280,  62,   5,   1],
       [ 68, 171,  59,  61],
       [  3,  40,  41, 262],
       [  0,   0,   4, 343]], dtype=int64)
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Apellidos: Bernal Castillo	09/05/2022
	Nombre: Aldo Alberto	

Máquina de Vectores

Ahora se aplican las técnicas de Máquina de Vectores para poder predecir price_range. En la siguiente gráfica, se muestra prince_range con respecto a las columnas **px_height y px_width**, se decidió usar estas 2 columnas ya que, de acuerdo con la matriz de correlaciones, su importancia es relevante para determinar prince_range.

```
fig, ax = plt.subplots(figsize=(8,4))
im = ax.scatter(data.px_height ,data.px_width, data.price_range , c = data.price_range, cmap= "jet" );
fig.colorbar(im, ax=ax)
ax.set_title("Datos");
#plt.scatter(data.px_height ,data.px_width,data.price_range , c = data.price_range )
```



En seguida, ya con la Base de Datos cargada en el DataFrame, se empieza a crear el modelo y los datos de entrenamiento. Empezamos con los datos de entrenamiento para X y Y.

```
# División de los datos en train y test
# ======================================================================
X = df.drop(columns = 'price_range')
y = df['price_range']

X_train, X_test, y_train, y_test = train_test_split(
                                        X,
                                        y.values.reshape(-1,1),
                                        train_size   = 0.8,
                                        random_state = 1234,
                                        shuffle      = True
                                    )
```

Ahora se empezará a entrenar el modelo con los datos de X y Y , además guardamos las predicciones del modelo en una variable para poder hacer su análisis.

```
# Creación del modelo SVM lineal
# ======================================================================
modelo = SVC(C = 100, kernel = 'linear', random_state=123)
modelo.fit(X_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d a
rray was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)


SVC(C=100, kernel='linear', random_state=123)
```

```
predicciones = modelo.predict(X_test)
```

Evaluación del modelo de Máquina de Vectores.

A continuación, se muestran varias métricas para la evaluación del algoritmo, usaremos los siguientes métodos, previamente definidos en el documento.

Accuracy, Precisión, Recall y F1 score

```
# Classification report
print(classification_report(y_true= y_test,y_pred=predicciones))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99        93
           1       0.96      0.94      0.95       102
           2       0.93      0.96      0.94        98
           3       0.99      0.98      0.99       107

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

Se puede observar que se tiene una precisión muy alta a diferencia de la Red Neuronal, la más alta fue de .99 teniendo un promedio de **.97**. Es decir, cuando el modelo predice la clase positiva ¿Cuántas veces está en lo cierto? lo cual indica que **los verdaderos positivos del modelo son altos**.

Como se dijo anteriormente en el análisis de la Red Neuronal, F1 es la **media entre la precisión y el recall,** simplifica el rendimiento a una única métrica, podemos observar la tendencia al alta mientras más se hace un Score.

Matriz de confusión del modelo Máquina de Vectores

Como se indicó anteriormente en el presente documento, la categoría de interés se conoce como **clase positiva**. A continuación, se muestran 2 matrices de confusión, la primera es la que el programa nos muestra al momento de correr el código, y la segunda muestra la lectura que se hace como **True Positive** de las instancias expuestas. Un detalle importante es que se muestran valores más pequeños que la Red Neuronal, esto es debido a que las métricas de rendimiento son tan altas que la matriz de confusión mide y **detalla los pocos datos a negociar como True Positive y False Negative.**

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_true= y_test,y_pred=predicciones)
```

```
array([[ 92,   1,   0,   0],
       [  1,  96,   5,   0],
       [  0,   3,  94,   1],
       [  0,   0,   2, 105]], dtype=int64)
```

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_true= y_test,y_pred=predicciones)
```

```
array([[ 92,   1,   0,   0],
       [  1,  96,   5,   0],
       [  0,   3,  94,   1],
       [  0,   0,   2, 105]], dtype=int64)
```

Conclusiones

Se utilizaron 2 modelos diferentes para poder obtener el price_range de la Base de Datos, Redes Neuronales y Máquina de Vectores. Al momento de realizar la medición de métricas para ambos modelos, podemos observar que la **Máquina de Vectores nos da una alta fidelidad para los resultados**. **Cabe resaltar que para el modelo de Red Neuronal solamente se uso 1 capa,** es posible que usando diferentes Layers y neuronas podamos obtener un resultado mucho más preciso.

Bibliografía

*https://www.datasource.ai/es/data-science-articles/comprension-de-la-matriz-de-confusion-y-como-implementarla-en-python

*https://empresas.blogthinkbig.com/como-interpretar-la-matriz-de-confusion-ejemplo-practico/