Proyecto 2 Iris Species

Tuesday, April 28, 2020 5:44 PM

Este proyecto es un clasificador para las flores de tipo Iris, el Dataset se obtiene de la página de kaggle en la cual tu deberás tener una cuenta para descargarlos, el enlace para obtenerlos es el siguiente: https://www.kaggle.com/uciml/iris

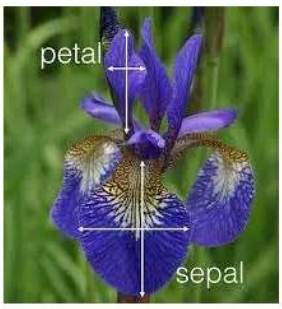
El Dataset contiene un total de 50 datos por cada especie de flor Iris, está se divide en 3 especies

- Iris Versicolor
- Iris Setosa
- Iris Virginica

Para cada una de las especies se midieron 4 características importantes:

- Longitud del sépalos
- Ancho del sépalos
- Longitud del pétalo
- Ancho del pétalo

Conociendo estos detalles se comenzará el proyecto en Python



El archivo con nombre "Iris.csv" es con el que vamos a trabajar y cuenta con 6 columnas que son:

- Id
- SepalLengthCm
- SepalWidthCm
- PetalLengthCm
- PetalWidthCm
- Species

	A	B	C	D	E	F
1	Id	SepalLength	SepalWidthC	PetalLengthC	PetalWidthC	Species
2	1	5.1	3.5	1.4	0.2	Iris-setosa
3	2	4.9	3	1.4	0.2	Iris-setosa
4	3	4.7	3.2	1.3	0.2	Iris-setosa
5	4	4.6	3.1	1.5	0.2	Iris-setosa
6	5	5	3.6	1.4	0.2	Iris-setosa
7	6	5.4	3.9	1.7	0.4	Iris-setosa
8	7	4.6	3.4	1.4	0.3	Iris-setosa

Para iniciar con la programación se debe contar con las siguientes 4 librerías instaladas

- Scikit-Learn
- Pandas
- Seaborn
- Matplotlib

Estás últimas 2 son para realizar gráficos

También se hará uso de 4 modelos de machine Learning para comprobar el mejor para el siguiente problema, los modelos a usar son:

- Regresión Logística
- Máquina de Vectores de Soporte
- Vecinos más cercanos
- Árbol de Decisiones

```
#|==================================|
#|Importamos las librerias a utilizar|
#|==================================|

import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
#Librerias de Machine Learning
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
```

Comenzaremos a leer el archivo con pandas, para ello crearemos la variable iris y vamos a imprimir las primeras 5 filas con la función head

```
iris=pd.read_csv('Iris.csv')
print(iris.head())
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

Se eliminará la columna Id ya que no se utilizara y se va a imprimir las primeras 5 filas nuevamente

```
#Se elimina la columna Id
iris=iris.drop("Id",axis=1)
print(iris.head())
```

```
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0            5.1           3.5            1.4           0.2  Iris-setosa
1            4.9           3.0            1.4           0.2  Iris-setosa
2            4.7           3.2            1.3           0.2  Iris-setosa
3            4.6           3.1            1.5           0.2  Iris-setosa
4            5.0           3.6            1.4           0.2  Iris-setosa
```

Vamos a analizar la información del Data iris, el tipo de dato, cuantos valores no nulos tenemos en las columnas y se imprime la información

```
#Analizamos los datos que tenemos disponibles
print("\nInformación del Dataset: ")
print(iris.info())
```

```
Información del Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 5.3+ KB
None
```

De la información mosrada lo más relevante es que las primeras 4 columnas manejan datos flotantes y la ultima de objeto, por lo cual ahora vamos a describir la información del Dataset, para lo cual se usara el siguiente comando

```
#Describimos la información del Dataset
print("\nDescripción del Dataset: ")
print(iris.describe())
```

```
Descripción del Dataset:
       SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count     150.000000    150.000000     150.000000    150.000000
mean        5.843333      3.054000       3.758667      1.198667
std         0.828066      0.433594       1.764420      0.763161
min         4.300000      2.000000       1.000000      0.100000
25%         5.100000      2.800000       1.600000      0.300000
50%         5.800000      3.000000       4.350000      1.300000
75%         6.400000      3.300000       5.100000      1.800000
max         7.900000      4.400000       6.900000      2.500000
```

La descripción del Dataset nos indica la cantidad de datos que hay en cada columna, la media, su desviación estándar y demás daros estadísticos

Ahora, vamos a verificar la distribución de los datos según la especie con 2 comandos muy simples

```
#Se verifica la distribución de los datos segun la especie
print("\nDistribución de las especies de Iris: ")
print(iris.groupby('Species').size())
```

```
Distribución de las especies de Iris:
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

Se cuenta con 50 datos para cada especie de Iris, para entender la información de mejor forma, usaremos la librería matplotlib para graficar el Dataset.

Para esto se usara la función subplot donde tendremos un matriz de 1x2, 1 fila con 2 columnas para imprimir el grafico, para iniciar comenzaremos a graficar el tamaño del sépalo con las siguientes características:

- Topo de gráfico: Scatter
- Eje X: "SepalLengthCm"
- Eje Y: "SepalWidthCm"
- Ax: ax1
- Los colores irán de la siguiente manera
    - Iris Setosa => Azul
    - Iris Versicolor => Verde
    - Iris Virginica => Rojo

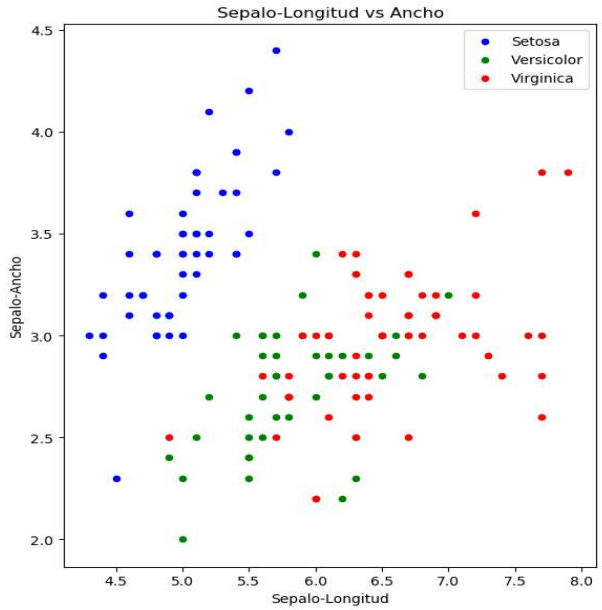
```
#(tipo de grafico,datos en eje x,datos en eje y, color del grafico,leyenda)
fig,(ax1,ax2)=plt.subplots(1,2)
iris[iris.Species=='Iris-setosa'].plot(kind='scatter',x='SepalLengthCm',y='SepalWidthCm',color='blue',label='Setosa',ax=ax1)
iris[iris.Species=='Iris-versicolor'].plot(kind='scatter',x='SepalLengthCm',y='SepalWidthCm',color='green',label='Versicolor',ax=ax1)
iris[iris.Species=='Iris-virginica'].plot(kind='scatter',x='SepalLengthCm',y='SepalWidthCm',color='red',label='Virginica',ax=ax1)
ax1.set_xlabel('Sepalo-Longitud')
ax1.set_ylabel('Sepalo-Ancho')
ax1.set_title('Sepalo-Longitud vs Ancho')
```

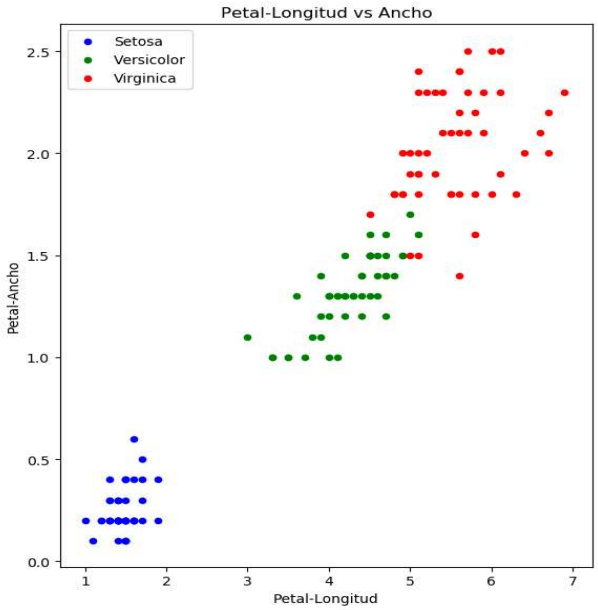
Continuamos con la información pétalo con las siguientes características:

- Topo de gráfico: Scatter
- Eje X: "PetalLengthCm"
- Eje Y: "PetalWidthCm"
- Ax: ax2
- Los colores irán de la siguiente manera
    - Iris Setosa => Azul
    - Iris Versicolor => Verde
    - Iris Virginica => Rojo

Mostraremos el grafico

```
#(tipo de grafico,datos en eje x,datos en eje y, color del grafico,leyenda)
iris[iris.Species=='Iris-setosa'].plot(kind='scatter',x='PetalLengthCm',y='PetalWidthCm',color='blue',label='Setosa',ax=ax2)
iris[iris.Species=='Iris-versicolor'].plot(kind='scatter',x='PetalLengthCm',y='PetalWidthCm',color='green',label='Versicolor',ax=ax2)
iris[iris.Species=='Iris-virginica'].plot(kind='scatter',x='PetalLengthCm',y='PetalWidthCm',color='red',label='Virginica',ax=ax2)
ax2.set_xlabel('Petal-Longitud')
ax2.set_ylabel('Petal-Ancho')
ax2.set_title('Petal-Longitud vs Ancho')
plt.show()
```





Del primer grafico correspondiente al tamaño del Sépalo se cuenta con la información:

- Los valores de Iris Setosa se encuentra uniforme
- Los valores de Iris Versicolor y Virginica no son uniformes, por lo tanto la información se mezcla

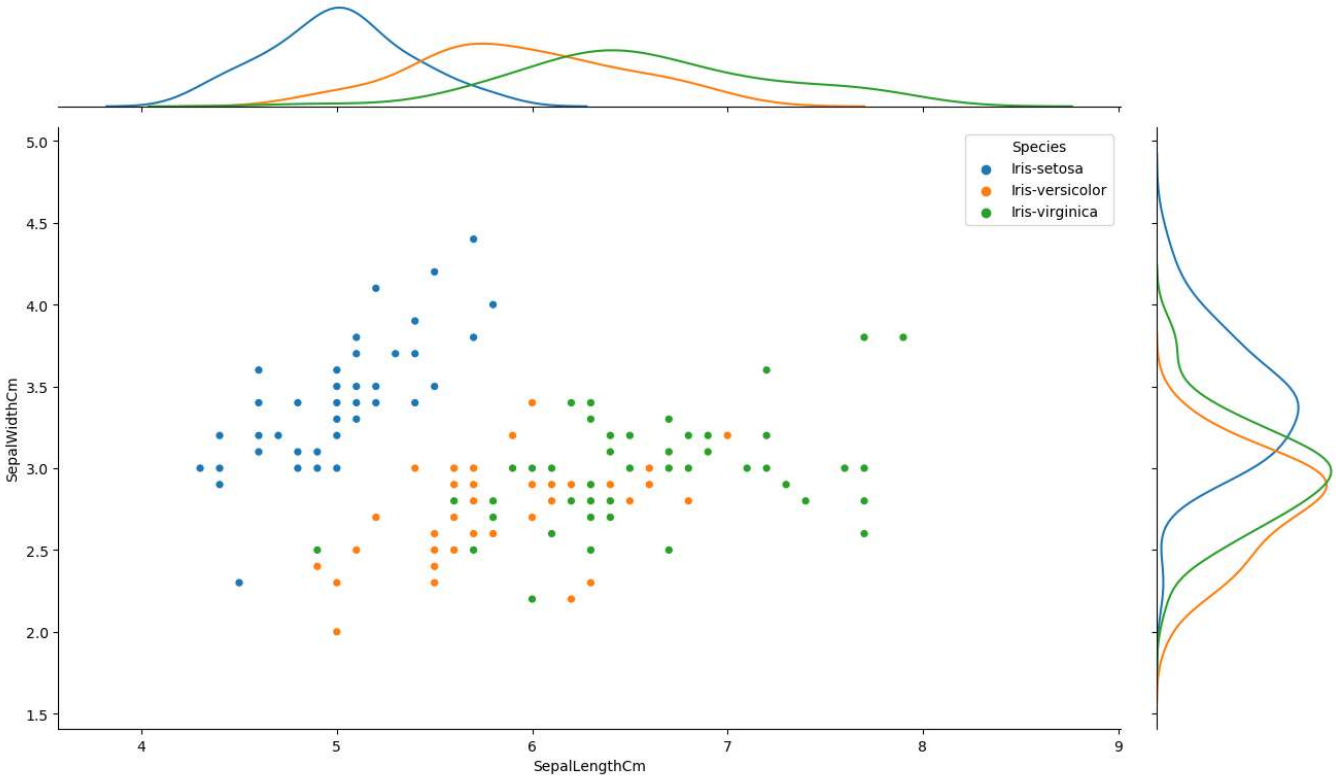
Del segundo gráfico correspondiente al tamaño del Pétalo se identifica lo siguiente:

- Los valores en las 3 especies de Iris son uniformes
- Los pétalos de Iris Setosa son los de menor tamaño
- Los pétalos de Iris Virginica son los de mayor tamaño

Esta información se hará de manera similar pero con la librería de Seaborne

```
#sns.jointplot(x='SepalLengthCm',y='SepalWidthCm',data=iris)
grid=sns.JointGrid(x='SepalLengthCm',y='SepalWidthCm',data=iris)
g=grid.plot_joint(sns.scatterplot,hue='Species',data=iris)
sns.kdeplot(iris.loc[iris['Species']=='Iris-setosa', 'SepalLengthCm'], ax=g.ax_marg_x, legend=False)
sns.kdeplot(iris.loc[iris['Species']=='Iris-versicolor', 'SepalLengthCm'], ax=g.ax_marg_x, legend=False)
sns.kdeplot(iris.loc[iris['Species']=='Iris-virginica', 'SepalLengthCm'], ax=g.ax_marg_x, legend=False)
sns.kdeplot(iris.loc[iris['Species']=='Iris-setosa', 'SepalWidthCm'], ax=g.ax_marg_y, vertical=True, legend=False)
sns.kdeplot(iris.loc[iris['Species']=='Iris-versicolor', 'SepalWidthCm'], ax=g.ax_marg_y, vertical=True, legend=False)
sns.kdeplot(iris.loc[iris['Species']=='Iris-virginica', 'SepalWidthCm'], ax=g.ax_marg_y, vertical=True, legend=False)
plt.show()
```

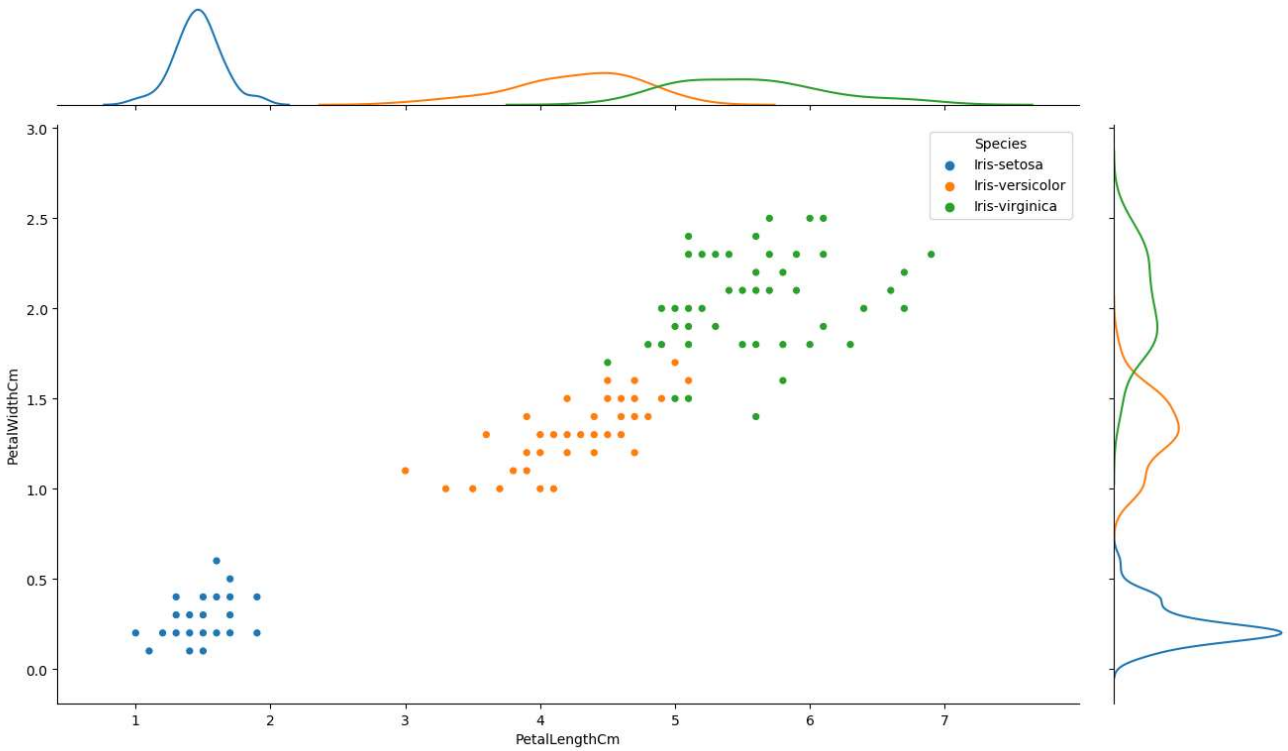
Este gráfico es similar al anterior con la diferencia que cuenta con un histograma que muestra la relación del tamaño del sépalo por especie tanto por su largo y el ancho



Se aplica el mismo procedimiento con el tamaño del pétalo

```
#sns.jointplot(x='PetalLengthCm',y='PetalWidthCm',data=iris)
grid=sns.JointGrid(x='PetalLengthCm',y='PetalWidthCm',data=iris)
g=grid.plot_joint(sns.scatterplot,hue='Species',data=iris)
sns.kdeplot(iris.loc[iris['Species']=='Iris-setosa', 'PetalLengthCm'], ax=g.ax_marg_x, legend=False)
sns.kdeplot(iris.loc[iris['Species']=='Iris-versicolor', 'PetalLengthCm'], ax=g.ax_marg_x, legend=False)
sns.kdeplot(iris.loc[iris['Species']=='Iris-virginica', 'PetalLengthCm'], ax=g.ax_marg_x, legend=False)
sns.kdeplot(iris.loc[iris['Species']=='Iris-setosa', 'PetalWidthCm'], ax=g.ax_marg_y, vertical=True, legend=False)
sns.kdeplot(iris.loc[iris['Species']=='Iris-versicolor', 'PetalWidthCm'], ax=g.ax_marg_y, vertical=True, legend=False)
sns.kdeplot(iris.loc[iris['Species']=='Iris-virginica', 'PetalWidthCm'], ax=g.ax_marg_y, vertical=True, legend=False)
plt.show()
```

Este gráfico es similar al anterior con la diferencia que cuenta con un histograma que muestra la relación del tamaño del pétalo por especie tanto por su largo y el ancho



Hasta el momento solo hemos analizado la información y hasta graficado de 2 maneras diferentes, el siguiente paso es el aplicar los modelos de Machine Learning, para lo cual, la información del Dataset iris se dividira en X que serán el ancho y largo del pétalo y sépalo y para la variable Y será la Especie.

También se dividirá el Dataset en un 80% de datos de entrenamiento y el 20% en datos de prueba

```
#Declaramos los valores X y Y
X=np.array(iris.drop(['Species'],1))
Y=np.array(iris['Species'])
#Separamos los datos de train y prueba para los algoritmos
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.2)
print('Son {} datos para entrenamiento y {} datos para la prueba'.format(X_train.shape[0],X_test.shape[0]))
```

El primer método a usar será la regresión logística y se va a imprimir la precisión del método

```
#------------------
#Regresión Logística
#------------------
RL=LogisticRegression()
RL.fit(X_train,y_train)
Y_pred_RL=RL.predict(X_test)
print('Precisión del algoritmo Regresión Logística es: {}'.format(RL.score(X_train,y_train)))
```

El segundo método a usar será el de máquinas de vectores de soporte y se va a imprimir la precisión del método

```
#------------------------------
#Máquinas de Vectores de Soporte
#------------------------------
MSV=SVC()
MSV.fit(X_train,y_train)
Y_pred_MSV=MSV.predict(X_test)
print('Precisión del algoritmo Máquinas de Vectores de Soporte es: {}'.format(MSV.score(X_train,y_train)))
```

El tercer método a usar será el de vecinos más cercanos y el número de vecinos será de 10 y se va a imprimir la precisión del método

```
#-------------------
#Vecinos más Cercanos
#-------------------
VC=KNeighborsClassifier(n_neighbors=10)
VC.fit(X_train,y_train)
Y_pred_VC=VC.predict(X_test)
print('Precisión del algoritmo Vecinos más Cercanos es: {}'.format(VC.score(X_train,y_train)))
```

Por último el cuarto método a usar será el árbol de decisiones y se va a imprimir la precisión del método

```
#-------------------
#Arbol de Decisiones
#-------------------
AD=DecisionTreeClassifier()
AD.fit(X_train,y_train)
Y_pred_AD=AD.predict(X_test)
print('Precisión del algoritmo Árbol de Decisiones es: {}'.format(AD.score(X_train,y_train)))
```

Una vez que se ejecuta el programa, se cuenta con los siguientes resultados

```
Son 120 datos para entrenamiento y 30 datos para la prueba
Precisión del algoritmo Regresión Logística es: 0.975
Precisión del algoritmo Máquinas de Vectores de Soporte es: 0.9583333333333334
Precisión del algoritmo Vecinos más Cercanos es: 0.9666666666666667
Precisión del algoritmo Árbol de Decisiones es: 1.0
```

Como se puede observar el método que da un mejor resultado es el de Árbol de Decisiones, por lo cual es el que se recomienda usar para este problema, aunque se puede modificar las consideraciones de los demás métodos para obtener un resultado mucho mejor