U N I V E R S I D A D D E P I U R A

VIDA UNIVERSITARIA

Análisis de datos con Python Nivel 2

Grupo 4:

"Trabajo final"

INTEGRANTES: El resto se retiraron.

Chavez Labrin, Luis Alberto David

Portal Chuan, Joaquin Rodrigo

DOCENTE

Ing. Pedro Rotta Saavedra

Piura, 06 de febrero de 2022

Introducción

El ser humano ha pasado por diversos cambios con el transcurso del tiempo, pequeños y grandes avances que hoy en día nos permiten hacer y planear cosas que hasta hace unas décadas eran inimaginables para el hombre.

Desde el inicio de la era tecnológica, la programación ha sido clave para los logros y avances obtenidos, además, nuevos métodos, herramientas y lenguajes se han ido acoplando para facilitar y mejorar su utilización.

Nuestras capacidades físicas son limitadas, por lo que, requerimos del uso de herramientas externas para lograr múltiples objetivos; sin embargo, no todas las herramientas pueden ser utilizadas sin una orden previa, aquí es donde la programación toma un papel fundamental; y se ve necesario el aprendizaje y dominio de un lenguaje de programación.

Gracias a la programación, se ha conseguido desde la posibilidad de dar órdenes simples a un computador, hasta llevar al hombre a la luna en 1969 con Apolo XI.

Con el fin de seguir contribuyendo en un mundo que está en constante cambio, en este proyecto hemos decidido enfocar lo aprendido en el curso de Python-nivel 2 para analizar un base de datos de kaggle con el fin de predecir el cáncer de mama usando machine learning.

Análisis del problema

El cáncer de mama es un tipo de cáncer muy común y ocupa el puesto 2 en causas de muertes de cáncer en mujeres.

Con la base de datos(diagnóstico) de cáncer de mama de Wisconsin en la data Wisconsin Breast Cancer podemos desarrollar un clasificador que pueda ayudar a diagnosticar a los pacientes y predecir la probabilidad de que esté con un cáncer de mama.

Mediante algoritmos de Machine learning clasificaremos si un cáncer de mama es maligno o benigno.

DESARROLLO DEL CÓDIGO:

A continuación, se mostrará el código desarrollado por el grupo y en ella se encontrará la descripción del procedimiento:

```
[ ]  # Predicción del cancer de mama-Python-Grupo 4
```

```
#Importamos las librerías necesarias para la ejecución del código
import numpy
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[2]  #Procedemos a leer los datos del archivo
df=pd.read_csv("data.csv")
df.head()
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      569 non-null    int64
 1   diagnosis               569 non-null    int64
 2   radius_mean             569 non-null    float64
 3   texture_mean            569 non-null    float64
```

```
[4] #Retornamos todas las columnas que tengan valores nulos
    df.isna().sum()
```

```
id                          0
diagnosis                   0
radius_mean                 0
texture_mean                0
perimeter_mean              0
```

```
[7] #Para conocer el tamaño del conjunto de datos
    df.shape
```

```
(569, 33)
```

```
[8] #Quitamos la columna nula:
    df=df.dropna(axis=1)
```

```
[9] #Tamaño de datos nuevo luego de remover la columna nula:
    df.shape
```

```
(569, 32)
```

```
#Descripción del conjunto de datos:
df.describe()
```

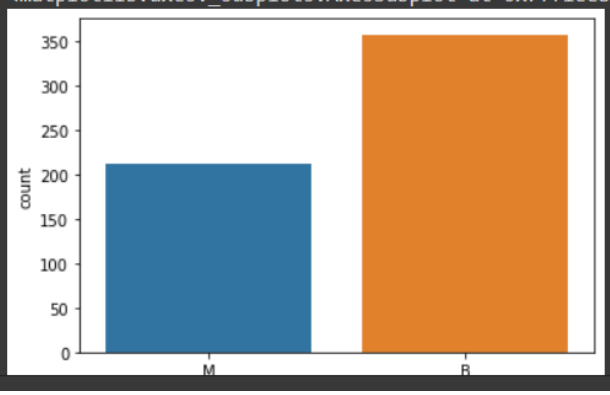
	id	radius_mean	texture_mean	perimeter_mean	area_mea
count	5.690000e+02	569.000000	569.000000	569.000000	569.00000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.88910
std	1.250206e+08	3.524049	4.301036	24.298981	351.91412
min	8.670000e+03	6.981000	9.710000	43.790000	143.50000
25%	8.692180e+05	11.700000	16.170000	75.170000	420.30000
50%	9.060240e+05	13.370000	18.840000	86.240000	551.10000

```
#Valor de células malignas y benignas: M(representa malignas) y B(representa benignas)
df['diagnosis'].value_counts()
```

```
B    357
M    212
Name: diagnosis, dtype: int64
```

```
#Análisis estadístico de celulas M y N:
sns.countplot(df['diagnosis'],label="count")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7ff1ee8ad090>
```

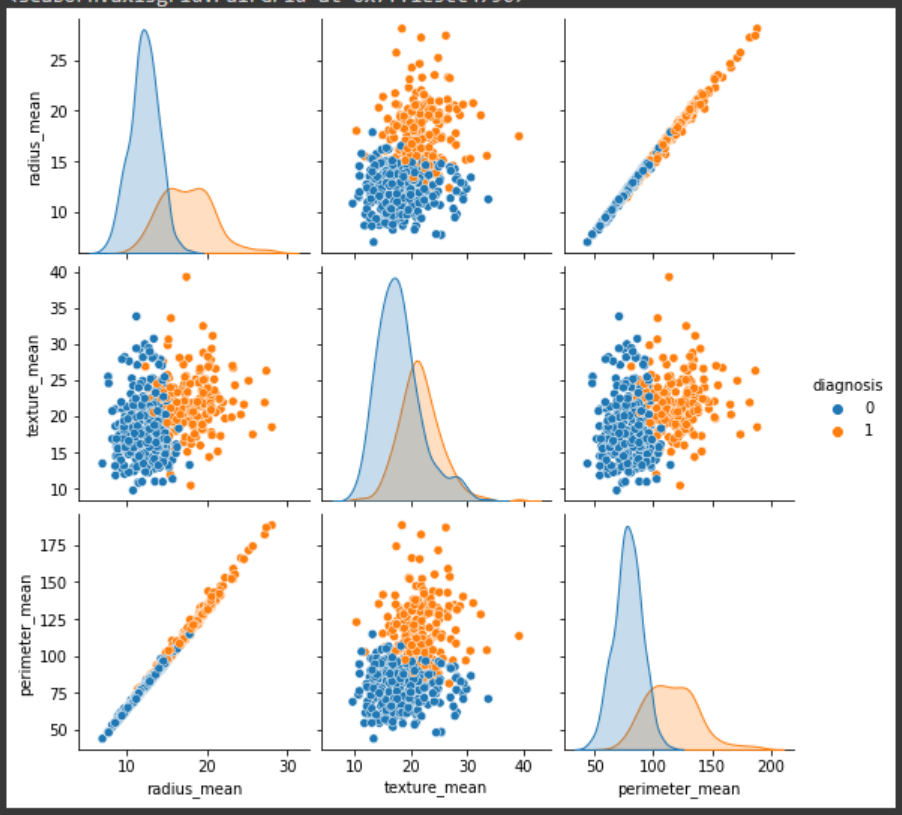



```
[18] #Convertimos los valores de M a 1 y B a 0:
     from sklearn.preprocessing import LabelEncoder
     labelencoder_Y = LabelEncoder()
     df.iloc[:,1]=labelencoder_Y.fit_transform(df.iloc[:,1].values)
     df.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_m
0	842302	1	17.99	10.38	122.80	10
1	842517	1	20.57	17.77	132.90	13
2	84300903	1	19.69	21.25	130.00	12
3	84348301	1	11.42	20.38	77.58	3
4	84358402	1	20.29	14.34	135.10	12

```
#Para observar unas gráficas de relación entre variables-Gráficas estadísticas y analíticas
sns.pairplot(df.iloc[:,1:5],hue="diagnosis")
```

```
<seaborn.axisgrid.PairGrid at 0x7ff1e3cc4790>
```

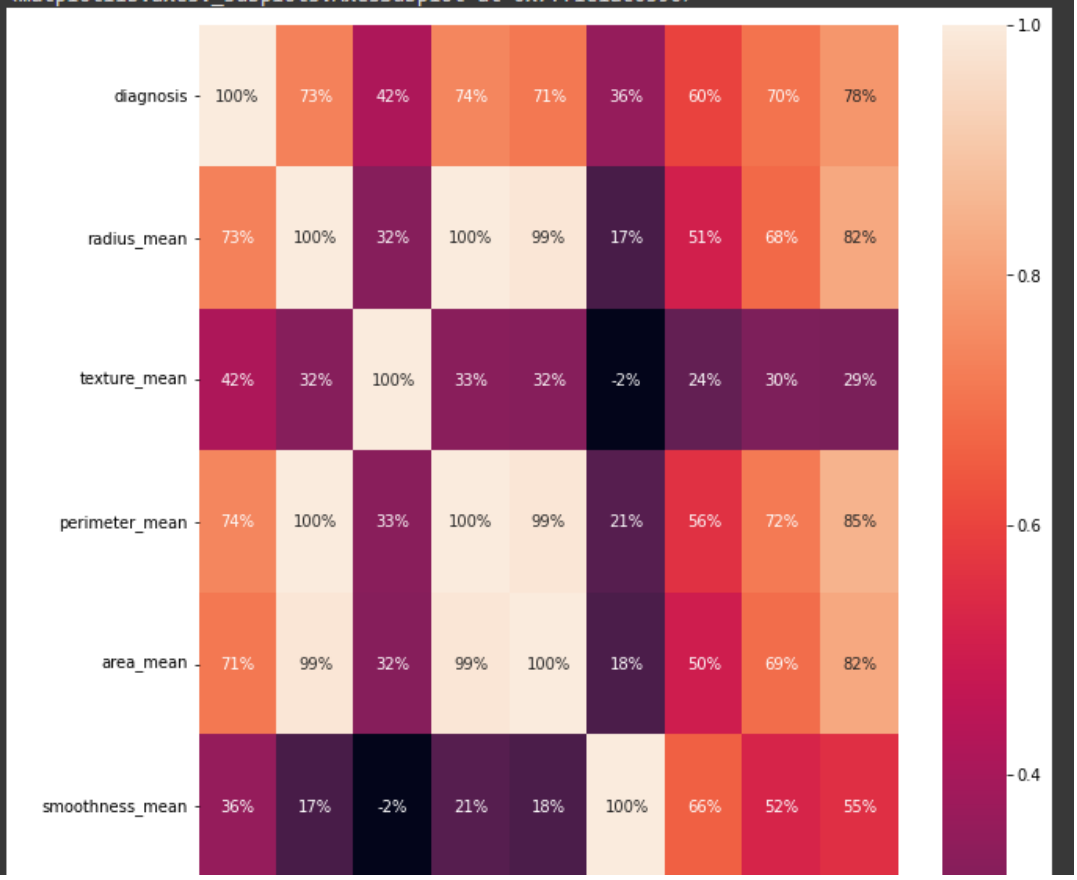



```
#Optenemos la correlacción que existe
df.iloc[:,1:32].corr()
```

	diagnosis	radius_mean	texture_mean	perimeter_mean
diagnosis	1.000000	0.730029	0.415185	0.742636
radius_mean	0.730029	1.000000	0.323782	0.997855
texture_mean	0.415185	0.323782	1.000000	0.329533
perimeter_mean	0.742636	0.997855	0.329533	1.000000
area_mean	0.708984	0.987357	0.321086	0.986507
smoothness_mean	0.358560	0.170581	-0.023389	0.207278
compactness_mean	0.596534	0.506124	0.236702	0.556936
concavity_mean	0.696360	0.676764	0.302418	0.716136
concave points_mean	0.776614	0.822529	0.293464	0.850977

```
#Visualizamos la correlación que existe-Correlación de Pearson
plt.figure(figsize=(10,15))
sns.heatmap(df.iloc[:,1:10].corr(),annot=True,fmt=".0%")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff1e1ac0390>
```




```
#Dividimos los datos en conjuntos independientes-X y dependientes-Y:
X=df.iloc[:,2:31].values
Y=df.iloc[:,1].values
#Dividimos la data en conjuntos de entrenamiento y de prueba del conjunto de datos:
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)
#Escalamiento de funciones:
from sklearn.preprocessing import StandardScaler
X_train=StandardScaler().fit_transform(X_train)
X_test=StandardScaler().fit_transform(X_test)
```

```python
#Modelos-se realizarán 3 algoritmos:
#Primer algoritmo: Regreción lógistica-Model 0
#Segundo modelo: Arbol de decisión-Model 1
#Tercer modelo: Bosque aleatorio_Model 2
#Se creará una función que contenga los 3 algoritmos:
def models(X_train,Y_train):
        #logistic regression
        from sklearn.linear_model import LogisticRegression
        log=LogisticRegression(random_state=0)
        log.fit(X_train,Y_train)


        #Decision Tree
        from sklearn.tree import DecisionTreeClassifier
        tree=DecisionTreeClassifier(random_state=0,criterion="entropy")
        tree.fit(X_train,Y_train)

        #Random Forest
        from sklearn.ensemble import RandomForestClassifier
        forest=RandomForestClassifier(random_state=0,criterion="entropy",n_estimators=10)
        forest.fit(X_train,Y_train)

        print('[0]logistic regression accuracy:',log.score(X_train,Y_train))
        print('[1]Decision tree accuracy:',tree.score(X_train,Y_train))
        print('[2]Random forest accuracy:',forest.score(X_train,Y_train))

        return log,tree,forest
```

```
[56] #Agregando los modelos en una variable para su posterior evaluación
     model=models(X_train,Y_train)
```

```
[0]logistic regression accuracy: 0.9912087912087912
[1]Decision tree accuracy: 1.0
[2]Random forest accuracy: 0.9978021978021978
```

```python
#Prueba de modelos
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

for i in range(len(model)):
    print("Model",i)
    print(classification_report(Y_test,model[i].predict(X_test)))
    print('Accuracy : ',accuracy_score(Y_test,model[i].predict(X_test)))
```

ANÁLISIS DEL RESULTADOS:

- Algoritmo de regresión logística: Se observa una precisión del 96.49%

```
Model 0
              precision    recall  f1-score   support

           0       0.96      0.99      0.97        67
           1       0.98      0.94      0.96        47

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114

Accuracy :  0.9649122807017544
```

```
[60] #Evaluación de predicción del algoritmo: regresión logística
     pred=model[0].predict(X_test)
     print('Valores predecidos:')
     print(pred)
     print('valores actuales:')
     print(Y_test)
```

```
Valores predecidos:
[1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0
 1 1 0]
valores actuales:
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
 1 1 0]
```

- Algoritmo de árbol de decisión: Se observa una precisión del 93.86%

```
Model 1
              precision    recall  f1-score   support

           0       0.94      0.96      0.95        67
           1       0.93      0.91      0.92        47

    accuracy                           0.94       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.94      0.94      0.94       114

Accuracy :  0.9385964912280702
```

```
#Evaluación de predicción del algoritmo: árbol de decisión
pred=model[1].predict(X_test)
print('Valores predecidos:')
print(pred)
print('Valores actuales:')
print(Y_test)
```

```
Valores predecidos:
[1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 1 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 0 0
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 0
 1 1 0]
Valores actuales:
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
 1 1 0]
```

- Algoritmo de bosque aleatorio: Se observa una precisión del 97.37%

```
Model 2
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        67
           1       1.00      0.94      0.97        47

    accuracy                           0.97       114
   macro avg       0.98      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114

Accuracy :  0.9736842105263158
```

```
#Evaluación de predicción del algoritmo: bosque aleatorio
pred=model[2].predict(X_test)
print('Valores predecidos:')
print(pred)
print('Valores actuales:')
print(Y_test)
```

```
Valores predecidos:
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 0
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
 1 1 0]
Valores actuales:
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
 1 1 0]
```

Conclusiones

- Se puede concluir que el algoritmo “bosque aleatorio” tiene una mejor precisión que los algoritmos “regresión logística” y “árbol de decisión”.
- A partir de lo desarrollado anteriormente, se observa que mientras mayor es la cantidad de datos, se puede obtener un mejor análisis y resultados para los mismos.
- Observándose que se puede usar machine learning para predecir el cáncer de mama, esto solo es un ejemplo de las diferentes formas en las que se puede emplear machine learning para casos y situaciones de la vida real.
- Finalizando con el trabajo, se observó que teniendo claro cuáles son las características a estudiar, o más conocidos como target, se pueden clasificar las características de un conjunto de datos para posteriormente poder hacer un análisis estadístico y predictivo de estos datos.
- Aunque no hemos podido concluir un análisis más profundo probando con más algoritmos, es entendible que existen aún mejores modelos y que posiblemente pueden adaptarse mejor con más presión. Esto ayudaría a volver más eficiente nuestro algoritmo.