

---

# Comparación de modelos: pregunta 4 - primer parcial

— Brandon Francisco Hernández Troncoso —

---

# Red neuronal artificial

- 4 capas: capa de entrada, 4 neuronas, 9 neuronas y capa de salida
- Activación: relu
- Learning\_rate: 0.1
- Solver: SGD

```
In [105]: runfile('C:/Users/brand/
Aprendizaje-de-maquina/Examen par
NN score: 0.9700000000000001

In [106]: runfile('C:/Users/brand/
Aprendizaje-de-maquina/Examen par
NN score: 0.9566666666666667

In [107]: runfile('C:/Users/brand/
Aprendizaje-de-maquina/Examen par
NN score: 0.99

In [108]: |
```

Se genera un promedio del  
Score con 10 instancias  
distintas del modelo

# SVM

- Kernel: lineal
- Penalización: 0.9

```
In [113]: runfile('C:/Users/brand/Documents/Aprendizaje-de-maquina/Examen parcial')
SVM score: 0.9700000000000001

In [114]: runfile('C:/Users/brand/Documents/Aprendizaje-de-maquina/Examen parcial')
SVM score: 0.9700000000000001

In [115]: runfile('C:/Users/brand/Documents/Aprendizaje-de-maquina/Examen parcial')
SVM score: 0.9666666666666666

In [116]:
```

ready Kite: ready conda:

- Kernel: poly
- Penalización: 0.9

```
In [110]: runfile('C:/Users/brand/Documents/Aprendizaje-de-maquina/Examen parcial')
SVM score: 0.9066666666666666

In [111]: runfile('C:/Users/brand/Documents/Aprendizaje-de-maquina/Examen parcial')
SVM score: 0.9233333333333335

In [112]: runfile('C:/Users/brand/Documents/Aprendizaje-de-maquina/Examen parcial')
SVM score: 0.9133333333333334

In [113]:
```

ready Kite: ready conda: python36 (

Se genera un promedio del Score con 10 instancias distintas del modelo

# KNN

- Neighbors: 3

```
In [113]: runfile('C:/Users/bran  
Aprendizaje-de-maquina/Examen pa  
SVM score: 0.9700000000000001  
  
In [114]: runfile('C:/Users/bran  
Aprendizaje-de-maquina/Examen pa  
SVM score: 0.9700000000000001  
  
In [115]: runfile('C:/Users/bran  
Aprendizaje-de-maquina/Examen pa  
SVM score: 0.9666666666666666
```

- Neighbors: 5

```
In [119]: runfile('C:/Users/brand/Documents  
Aprendizaje-de-maquina/Examen parcial')  
K Neighbors score: 0.95  
  
In [120]: runfile('C:/Users/brand/Documents  
Aprendizaje-de-maquina/Examen parcial')  
K Neighbors score: 0.96  
  
In [121]: runfile('C:/Users/brand/Documents  
Aprendizaje-de-maquina/Examen parcial')  
K Neighbors score: 0.9400000000000001  
  
In [122]:  
  
n: ready Kite: ready conda: python36
```

Se genera un promedio del  
Score con 10 instancias  
distintas del modelo

# Naive Bayes

```
Aprendizaje-de-maquina/Examen parcial')  
Naive Bayes: 0.9700000000000001  
  
In [123]: runfile('C:/Users/brand/Documentos/Aprendizaje-de-maquina/Examen parcial')  
Naive Bayes: 0.9566666666666667  
  
In [124]: runfile('C:/Users/brand/Documentos/Aprendizaje-de-maquina/Examen parcial')  
Naive Bayes: 0.9733333333333334  
  
In [125]:  
  
n: ready Kite: ready conda: python
```

Se genera un promedio del Score con 10 instancias distintas del modelo

# Conclusión

Se puede observar que al ser un dataset de “juguete”, casi todos los modelos dan un resultado de score muy parecido, sin embargo el modelo que presenta menos variación observada con las pruebas fue el de SVM con kernel lineal. En seguida, se puede observar que el segundo modelo más preciso es KNN con 3 vecinos ya que las ANN tienen un score muy cercano a KNN pero con más variación. Por último, el modelo menos acertado es NB, sin embargo sigue teniendo un score muy alto.

# Códigos ANN y SVM

## ANN

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 @author: brandonhdz
5 """
6 import matplotlib.pyplot as plt
7 from sklearn import neural_network, model_selection, preprocessing
8 from sklearn.datasets import load_iris
9
10 X, y=load_iris(return_X_y=True)
11
12 score=0
13 for i in range(10):
14     X_train, X_test, y_train, y_test=model_selection.train_test_split(X,y,test_size=0.2)
15
16     scaler=preprocessing.StandardScaler()
17     scaler.fit(X_train)
18     X_train=scaler.transform(X_train)
19     X_test=scaler.transform(X_test)
20
21
22     modelo=neural_network.MLPClassifier(
23         hidden_layer_sizes=(4,9),
24         activation='relu',
25         learning_rate_init=0.1,
26         max_iter=1000,
27         solver='sgd'
28     )
29
30     modelo.fit(X_train, y_train)
31
32     score+=modelo.score(X_test, y_test)
33
34 score=score/10
35
36 print("NN score: ",score)
```

## SVM

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: brandonhdz
"""
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_iris
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

X, y = load_iris(return_X_y=True)

score=0
for i in range(10):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

    scaler=StandardScaler()
    scaler.fit(X_train)
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)

    modelo=SVC(kernel='Linear', C=0.9)

    modelo.fit(X_train, y_train)

    score+=modelo.score(X_test, y_test)

score=score/10

print("SVM score: ",score)
```

# Códigos KNN y NB

## KNN

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  @author: brandonhdz
5  """
6  from sklearn.preprocessing import StandardScaler
7  from sklearn.datasets import load_iris
8  from sklearn.model_selection import train_test_split
9  from sklearn import neighbors
10
11  X, y = load_iris(return_X_y=True)
12
13  score=0
14  for i in range(10):
15      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
16
17      scaler=StandardScaler()
18      scaler.fit(X_train)
19      X_train = scaler.transform(X_train)
20      X_test = scaler.transform(X_test)
21
22      n_neighbors=5
23
24      modelo=neighbors.KNeighborsClassifier(n_neighbors)
25
26      modelo.fit(X_train, y_train)
27
28      score+=modelo.score(X_test, y_test)
29
30  score=score/10
31
32  print("K Neighbors score: ", score)
```

## NB

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  @author: brandonhdz
5  """
6  from sklearn.preprocessing import StandardScaler
7  from sklearn.datasets import load_iris
8  from sklearn.model_selection import train_test_split
9  from sklearn.naive_bayes import GaussianNB
10
11  X, y = load_iris(return_X_y=True)
12
13  score=0
14  for i in range(10):
15
16      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
17
18      #Scaling data
19      scaler=StandardScaler()
20      scaler.fit(X_train)
21      X_train = scaler.transform(X_train)
22      X_test=scaler.transform(X_test)
23
24      modelo = GaussianNB()
25
26      modelo.fit(X_train, y_train)
27
28      score+=modelo.score(X_test, y_test)
29
30  score=score/10
31
32  print("Naive Bayes: ", score)
```