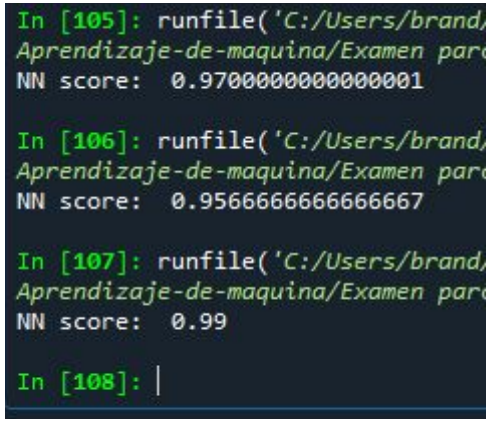
Comparación de modelos: pregunta 4 - primer parcial

Brandon Francisco Hernández Troncoso

Red neuronal artificial

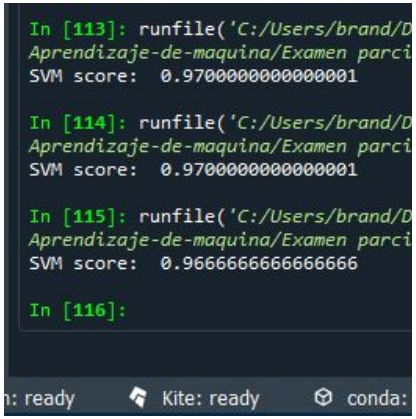
- 4 capas: capa de entrada, 4 neuronas, 9 neuronas y capa de salida
- Activación: relu
- Learning_rate: 0.1
- Solver: SGD



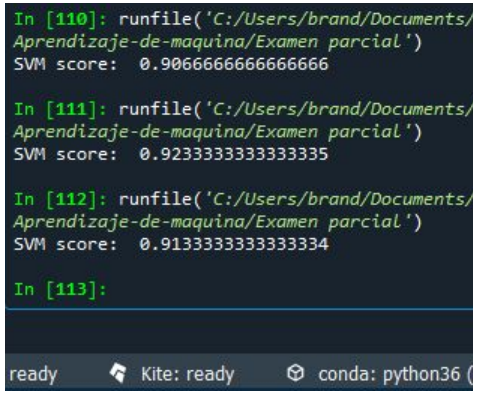
Se genera un promedio del Score con 10 instancias distintas del modelo

SVM

- Kernel: lineal
- Penalización: 0.9



Kernel: poly
Penalización: 0.9



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

KNN

- Neighbors: 3

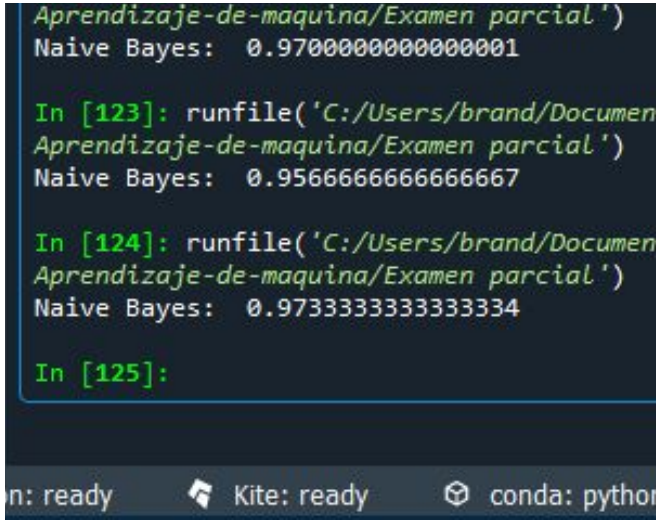
```
In [113]: runfile('C:/Users/bran
Aprendizaje-de-maquina/Examen pa
SVM score:  0.9700000000000001

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



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

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: brandonhdz
"""
import matplotlib.pyplot as plt
from sklearn import neural_network, model_selection, preprocessing
from sklearn.datasets import load_iris

X, y=load_iris(return_X_y=True)

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

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


    modelo=neural_network.MLPClassifier(
        hidden_layer_sizes=(4,9),
        activation='relu',
        learning_rate_init=0.1,
        max_iter=1000,
        solver='sgd'
        )

    modelo.fit(X_train, y_train)

    score+=modelo.score(X_test, y_test)

score=score/10

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

SVM

```python
#!/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

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

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)

    n_neighbors=5

    modelo=neighbors.KNeighborsClassifier(n_neighbors)

    modelo.fit(X_train, y_train)

    score+=modelo.score(X_test, y_test)

score=score/10

print("K Neighbors score: ", score)
```

NB

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

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)

    #Scaling data
    scaler=StandardScaler()
    scaler.fit(X_train)
    X_train = scaler.transform(X_train)
    X_test=scaler.transform(X_test)

    modelo = GaussianNB()

    modelo.fit(X_train, y_train)

    score+=modelo.score(X_test, y_test)

score=score/10

print("Naive Bayes: ", score)
```