Avances

Clasificadores

K Vecinos Más Cercanos

[376]:
```
#Precisión del modelo - Prueba
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
pred = knn.predict(X_test)
print(classification_report(y_test, pred))
```

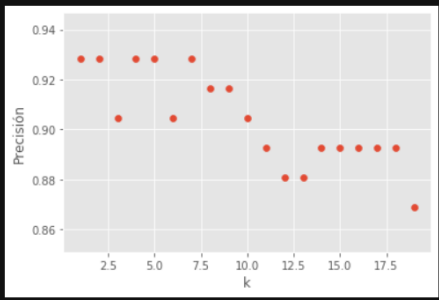
```
              precision    recall  f1-score   support

           0       1.00      0.60      0.75         5
           1       1.00      0.50      0.67         6
           2       0.33      0.40      0.36         5
           3       1.00      0.82      0.90        17
           4       0.89      1.00      0.94        25
           5       1.00      1.00      1.00         6
           6       0.75      0.90      0.82        20

    accuracy                           0.85        84
   macro avg       0.85      0.75      0.78        84
weighted avg       0.87      0.85      0.84        84
```

[380]:
```
#Precisión del modelo - Entrenamiento
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
pred = knn.predict(X_train)
print(classification_report(y_train, pred))
```

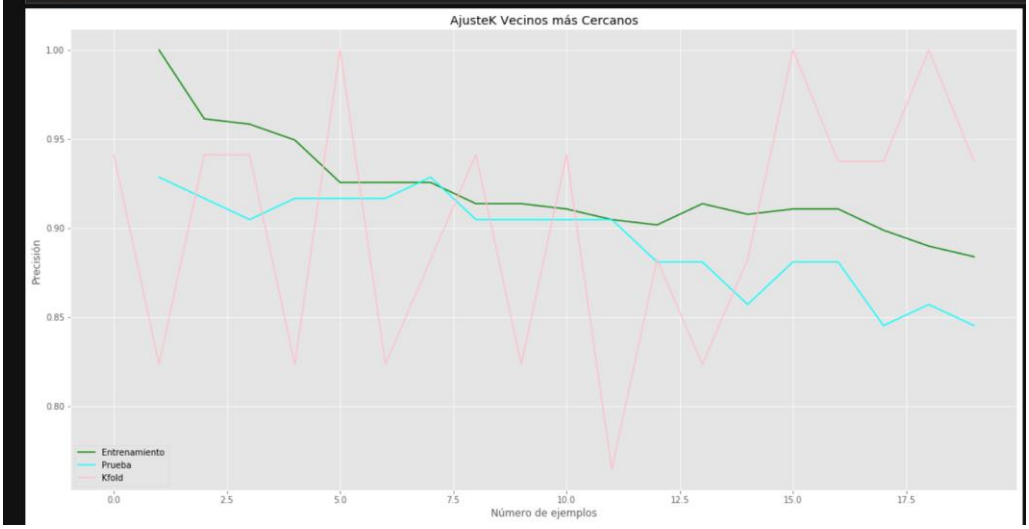
```
              precision    recall  f1-score   support

           0       0.67      0.36      0.47        22
           1       0.94      0.70      0.80        23
           2       0.58      0.58      0.58        26
           3       1.00      0.90      0.95        60
           4       0.93      1.00      0.96        93
           5       1.00      1.00      1.00        27
           6       0.77      0.91      0.83        85

    accuracy                           0.86       336
   macro avg       0.84      0.78      0.80       336
weighted avg       0.86      0.86      0.86       336
```

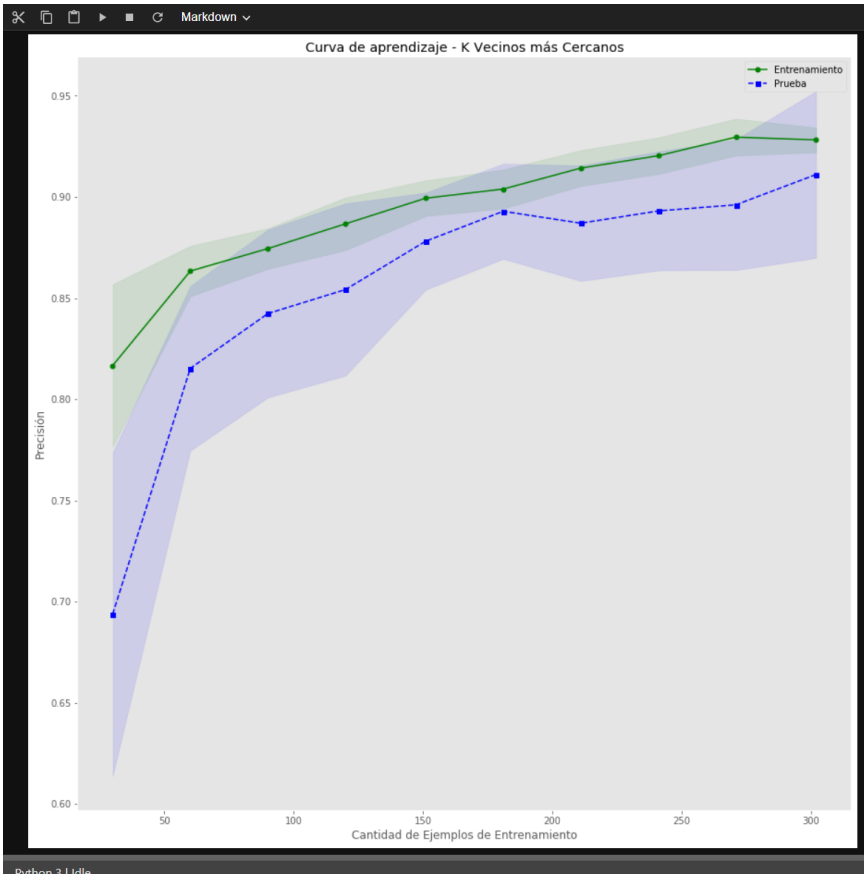
[423]:
```
#Elegir el mejor valor de k
k_range = range(1, numAttr)
scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train, y_train)
    scores.append(knn.score(X_test, y_test))
plt.figure()
plt.xlabel('k')
plt.ylabel('Precisión')
plt.scatter(k_range, scores)
```

[423]:
```
<matplotlib.collections.PathCollection at 0x1b6a6c5c4c8>
```



```
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train, y_train)
    train_prec.append(knn.score(X_train, y_train))
    eval_prec.append(knn.score(X_test, y_test))

# Resultados.
fig, ax = plt.subplots(figsize=(20, 10))
plt.plot(max_deep_list, train_prec, color="green", label="Entrenamiento")
plt.plot(max_deep_list, eval_prec, color="cyan", label="Prueba")
plt.plot(precisionKVCE, color="pink", label="Kfold")

plt.title('AjusteK Vecinos más Cercanos')
plt.legend()
plt.ylabel('Precisión')
plt.xlabel('Número de ejemplos')
plt.show()
```





Navie Bayes

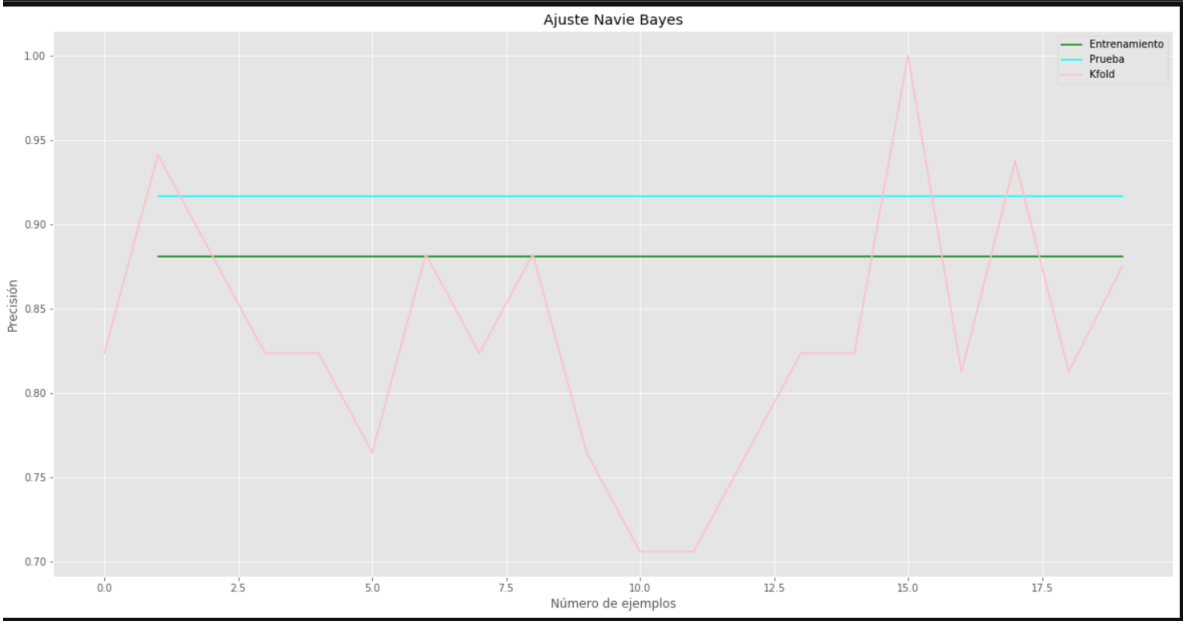
[390]:
```
#Precisión del modelo - Prueba
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

pred = gnb.predict(X_test)
print(classification_report(y_test, pred))
```

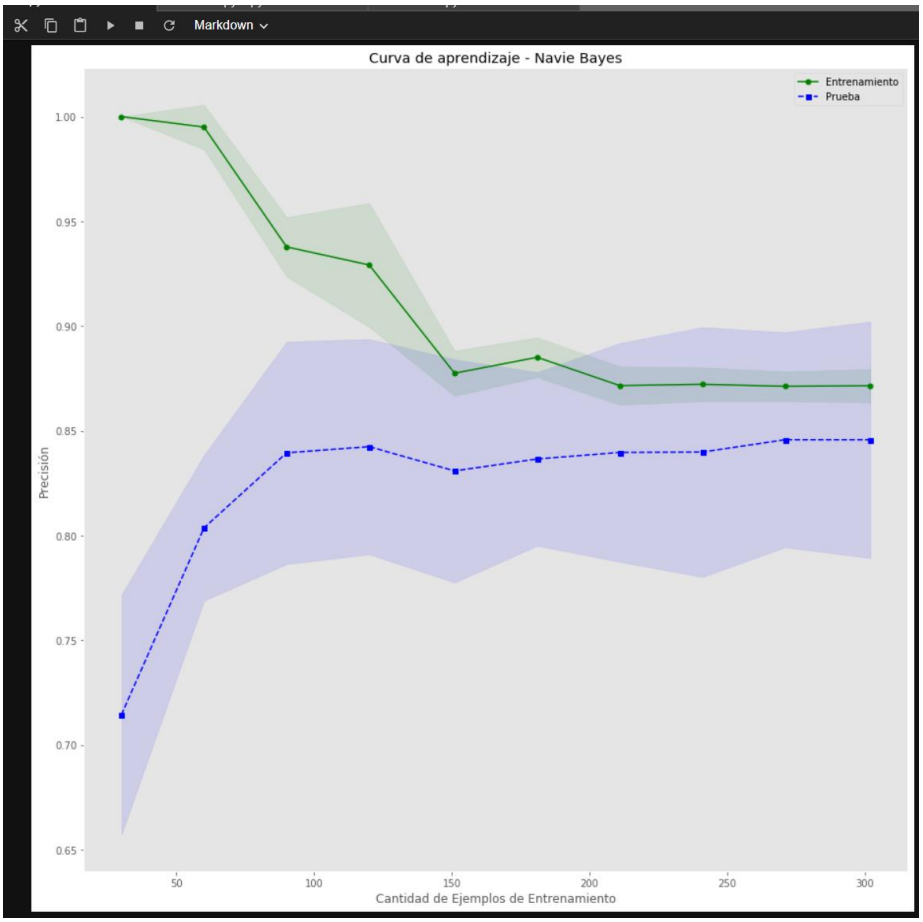
```
              precision    recall  f1-score   support

           0       0.70      1.00      0.82         7
           1       0.60      0.75      0.67         4
           2       1.00      0.33      0.50         6
           3       1.00      0.94      0.97        16
           4       1.00      0.93      0.96        28
           5       1.00      1.00      1.00         3
           6       0.83      0.95      0.88        20

    accuracy                           0.89        84
   macro avg       0.88      0.84      0.83        84
weighted avg       0.91      0.89      0.89        84
```

[391]:
```
#Precisión del modelo - Entrenamiento
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

pred = gnb.predict(X_train)
print(classification_report(y_train, pred))
```

```
              precision    recall  f1-score   support

           0       0.76      0.96      0.85        23
           1       0.72      0.88      0.79        26
           2       0.64      0.29      0.40        24
           3       1.00      0.98      0.99        62
           4       1.00      0.93      0.97        91
           5       1.00      1.00      1.00        27
           6       0.80      0.88      0.84        83

    accuracy                           0.89       336
   macro avg       0.85      0.85      0.83       336
weighted avg       0.89      0.89      0.88       336
```

```
    eval_prec.append(gnb.score(X_test, y_test))
    #precision((X_test, y_test))
# Resultados.
fig, ax = plt.subplots(figsize=(20, 10))
plt.plot(max_deep_list, train_prec, color="green", label="Entrenamiento")
plt.plot(max_deep_list, eval_prec, color="cyan", label="Prueba")
plt.plot(precisionGNBE, color="pink", label="Kfold")

plt.title("Ajuste Navie Bayes")
plt.legend()
plt.ylabel("Precisión")
plt.xlabel("Número de ejemplos")
plt.show()
```





Árbol de Decisión

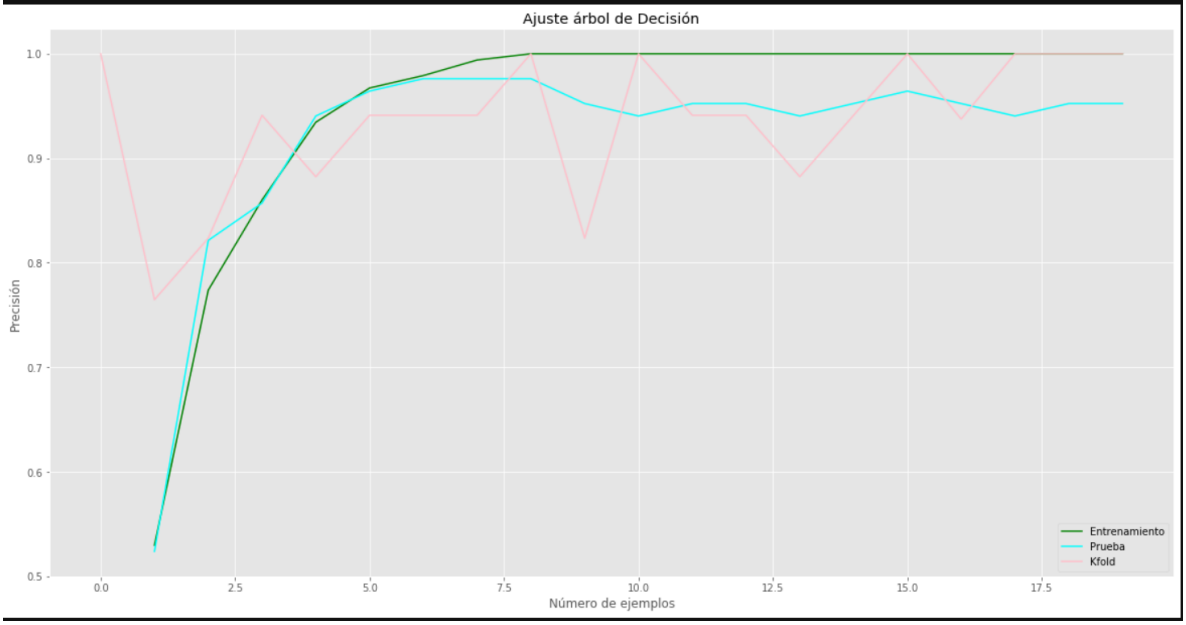
```
[405]: #Precisión del modelo - Prueba
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
ad = DecisionTreeClassifier(criterion = "entropy")
ad.fit(X_test, y_test)

pred = ad.predict(X_test)
print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	6
1	1.00	1.00	1.00	4
2	1.00	1.00	1.00	5
3	1.00	1.00	1.00	20
4	1.00	1.00	1.00	24
5	1.00	1.00	1.00	6
6	1.00	1.00	1.00	19
accuracy			1.00	84
macro avg	1.00	1.00	1.00	84
weighted avg	1.00	1.00	1.00	84

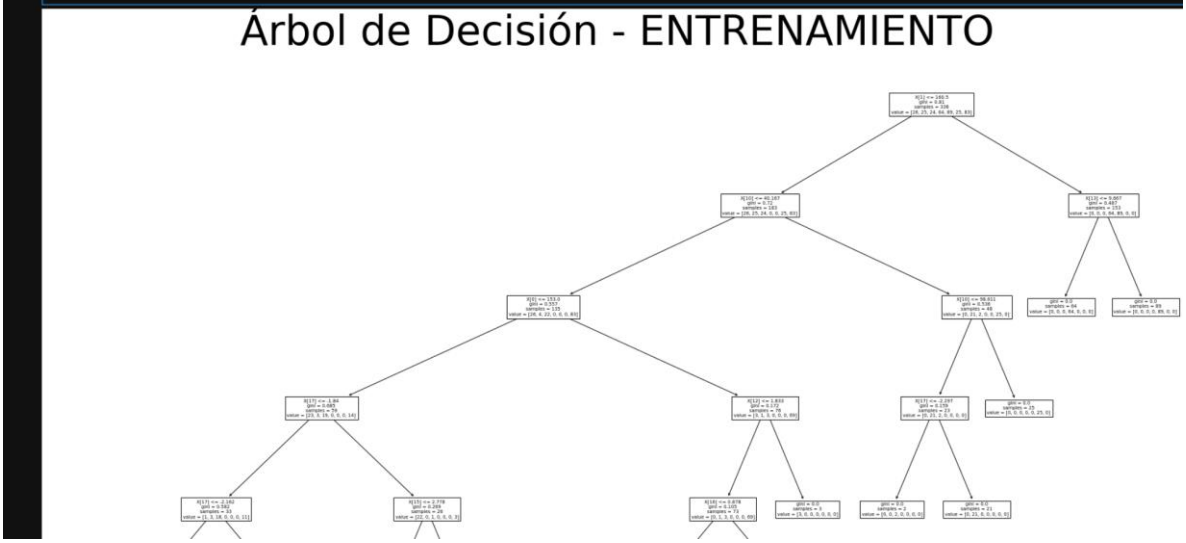
[406]:
```
#Precisión del modelo - Entrenamiento
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
ad = DecisionTreeClassifier(criterion = "entropy")
ad.fit(X_train, y_train)

pred = ad.predict(X_test)
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       0.67      0.50      0.57         4
           2       0.75      0.60      0.67         5
           3       1.00      1.00      1.00        20
           4       1.00      1.00      1.00        24
           5       1.00      1.00      1.00         6
           6       0.86      0.95      0.90        19

    accuracy                           0.94        84
   macro avg       0.90      0.86      0.88        84
weighted avg       0.94      0.94      0.94        84
```

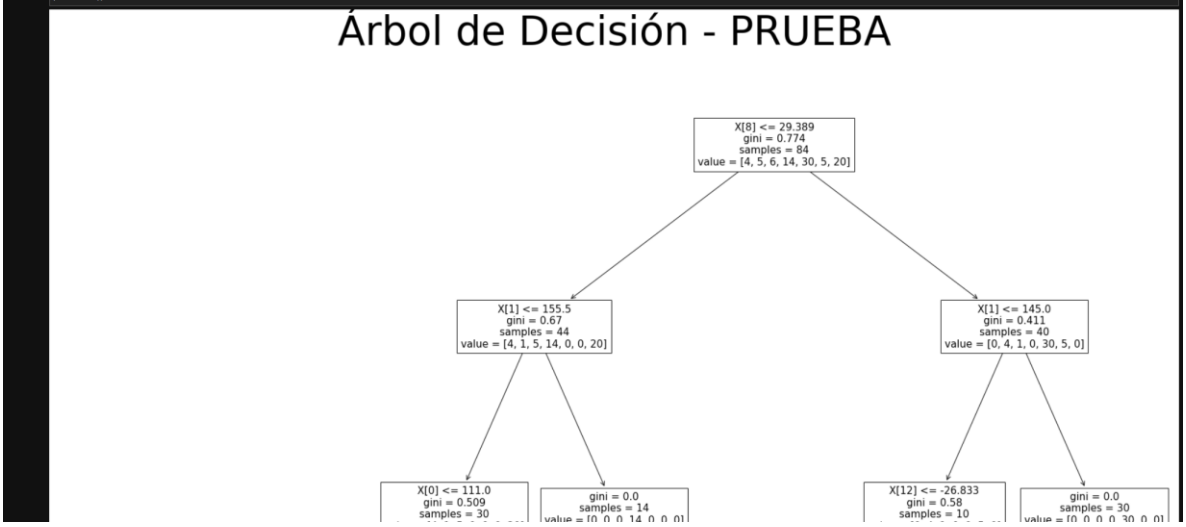
```
    arbol3.fit(X_train, y_train)
    train_prec.append(arbol3.score(X_train, y_train))
    eval_prec.append(arbol3.score(X_test, y_test))

# Resultados.
fig, ax = plt.subplots(figsize=(20, 10))
plt.plot(max_deep_list, train_prec, color="green", label="Entrenamiento")
plt.plot(max_deep_list, eval_prec, color="cyan", label="Prueba")
plt.plot(precisionADP, color="pink", label="Kfold")

plt.title("Ajuste árbol de Decisión")
plt.legend()
plt.ylabel("Precisión")
plt.xlabel("Número de ejemplos")
plt.show()
```



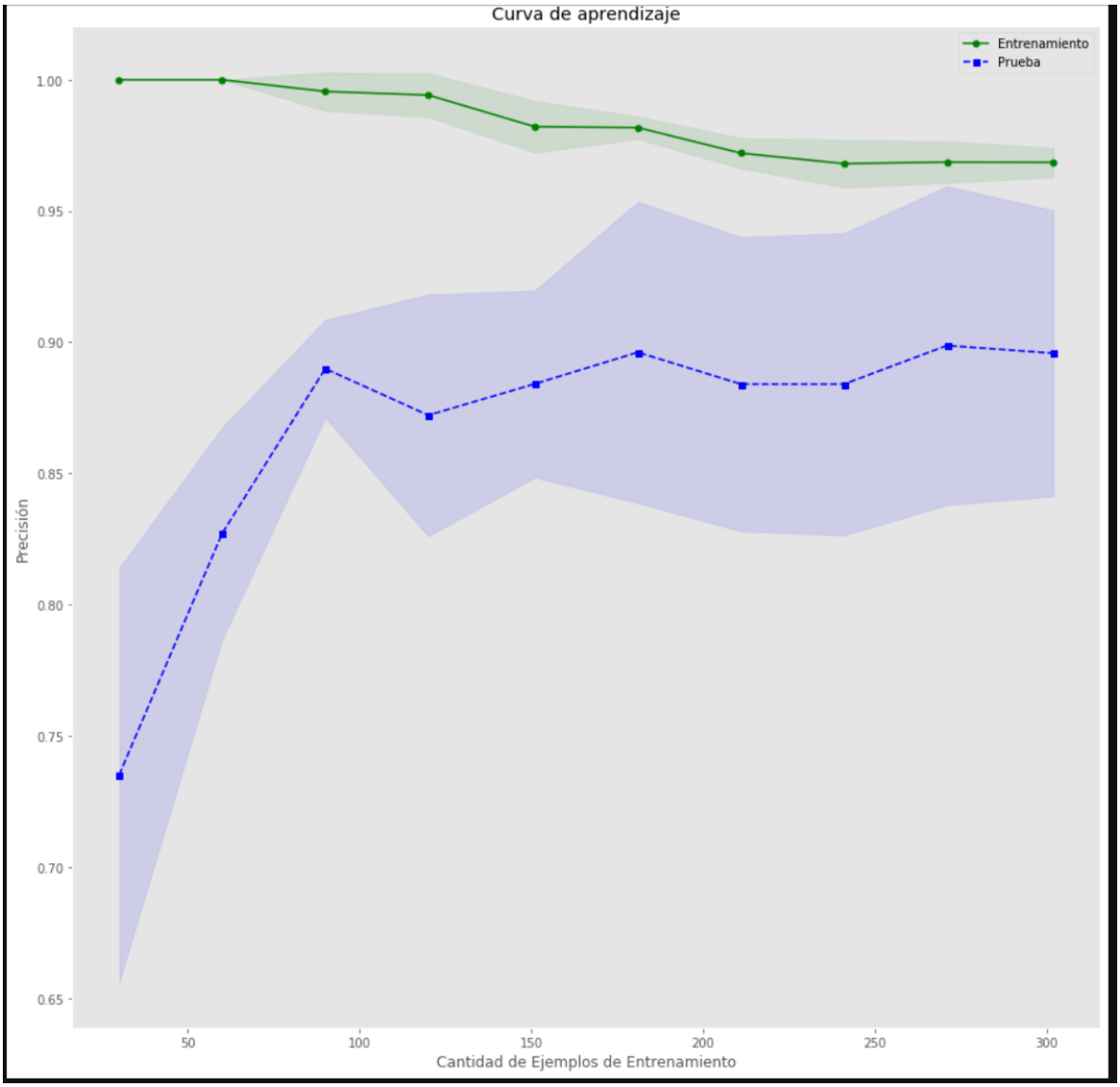
```
[100]: import matplotlib.pyplot as plt
       arbolito = tree.DecisionTreeClassifier()

       fig, ax = plt.subplots(figsize=(30, 30))

       tree.plot_tree(arbolito.fit(X_train, y_train))
       plt.title("Árbol de Decisión - ENTRENAMIENTO", fontsize=60)
       plt.show()
```




```
[114]: from sklearn import tree
       import matplotlib.pyplot as plt
       arbolito = tree.DecisionTreeClassifier()

       fig, ax = plt.subplots(figsize=(30, 30))

       tree.plot_tree(arbolito.fit(X_test, y_test))
       plt.title("Árbol de Decisión - PRUEBA", fontsize=60)
       plt.show()
```



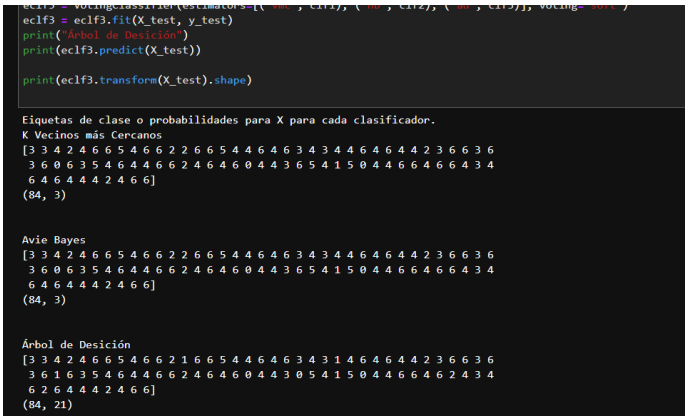
Curva de Aprendizaje



En este gráfico se puede ver claramente como con pocos datos la precisión entre los datos de entrenamiento y los de prueba son muy distintas y luego a medida que la cantidad de datos va aumentando, el modelo puede generalizar mucho mejor y las precisiones se comienzan a igualar. Este gráfico también puede ser importante a la hora de decidir invertir en la obtención de más datos, ya que por ejemplo nos indica que, a partir las 170 muestras, el modelo ya no gana mucha más precisión a pesar de obtener más datos.

Votación de los clasificadores

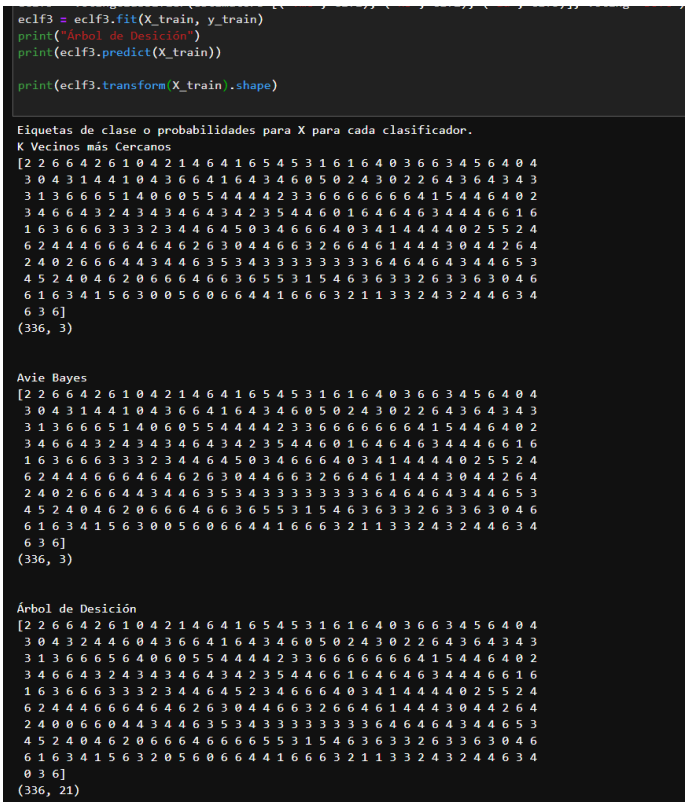
Prueba(“corrección”)



```
eclf3 = eclf3.fit(X_test, y_test)
print("Árbol de Desición")
print(eclf3.predict(X_test))

print(eclf3.transform(X_test).shape)
```

```
Etiquetas de clase o probabilidades para X para cada clasificador.
K Vecinos más Cercanos
[3 3 4 2 4 6 6 5 4 6 6 2 2 6 6 5 4 4 6 4 6 3 4 3 4 4 6 4 6 4 4 4 2 3 6 6 3 6
 3 6 0 6 3 5 4 6 4 4 6 6 2 4 6 4 6 0 4 4 3 6 5 4 1 5 0 4 4 6 6 4 6 6 4 3 4
 6 4 6 4 4 4 2 4 6 6]
(84, 3)


Avie Bayes
[3 3 4 2 4 6 6 5 4 6 6 2 2 6 6 5 4 4 6 4 6 3 4 3 4 4 6 4 6 4 4 4 2 3 6 6 3 6
 3 6 0 6 3 5 4 6 4 4 6 6 2 4 6 4 6 0 4 4 3 6 5 4 1 5 0 4 4 6 6 4 6 6 4 3 4
 6 4 6 4 4 4 2 4 6 6]
(84, 3)


Árbol de Desición
[3 3 4 2 4 6 6 5 4 6 6 2 1 6 6 5 4 4 6 4 6 3 4 3 1 4 6 4 6 4 4 2 3 6 6 3 6
 3 6 1 6 3 5 4 6 4 4 6 6 2 4 6 4 6 0 4 4 3 0 5 4 1 5 0 4 4 6 6 4 6 6 2 4 3 4
 6 2 6 4 4 4 2 4 6 6]
(84, 21)
```

Entrenamiento(“corrección”)



```
eclf3 = eclf3.fit(X_train, y_train)
print("Árbol de Desición")
print(eclf3.predict(X_train))

print(eclf3.transform(X_train).shape)
```

```
Etiquetas de clase o probabilidades para X para cada clasificador.
K Vecinos más Cercanos
[2 2 6 6 4 2 6 1 0 4 2 1 4 6 4 1 6 5 4 5 3 1 6 1 6 4 0 3 6 6 3 4 5 6 4 0 4
 3 0 4 3 1 4 4 1 0 4 3 6 6 4 1 6 4 3 4 6 0 5 0 2 4 3 0 2 2 6 4 3 6 4 3 4 3
 3 1 3 6 6 6 5 1 4 0 6 0 5 5 4 4 4 4 2 3 3 6 6 6 6 6 6 4 1 5 4 4 6 4 0 2
 3 4 6 6 4 3 2 4 3 4 3 4 6 4 3 4 2 3 5 4 4 6 0 1 6 4 6 4 6 3 4 4 4 6 6 1 6
 1 6 3 6 6 6 3 3 3 2 3 4 4 6 4 5 0 3 4 6 6 6 4 0 3 4 1 4 4 4 4 0 2 5 5 2 4
 6 2 4 4 4 6 6 6 4 6 4 6 2 6 3 0 4 4 6 6 3 2 6 6 4 6 1 4 4 4 3 0 4 4 2 6 4
 2 4 0 2 6 6 6 4 4 3 4 4 6 3 5 3 4 3 3 3 3 3 3 3 3 6 4 6 4 6 4 3 4 4 6 5 3
 4 5 2 4 0 4 6 2 0 6 6 6 4 6 6 3 6 5 5 3 1 5 4 6 3 6 3 3 2 6 3 3 6 3 0 4 6
 6 1 6 3 4 1 5 6 3 0 0 5 6 0 6 6 4 4 1 6 6 6 3 2 1 1 3 3 2 4 3 2 4 4 6 3 4
 6 3 6]
(336, 3)


Avie Bayes
[2 2 6 6 4 2 6 1 0 4 2 1 4 6 4 1 6 5 4 5 3 1 6 1 6 4 0 3 6 6 3 4 5 6 4 0 4
 3 0 4 3 1 4 4 1 0 4 3 6 6 4 1 6 4 3 4 6 0 5 0 2 4 3 0 2 2 6 4 3 6 4 3 4 3
 3 1 3 6 6 6 5 1 4 0 6 0 5 5 4 4 4 4 2 3 3 6 6 6 6 6 6 4 1 5 4 4 6 4 0 2
 3 4 6 6 4 3 2 4 3 4 3 4 6 4 3 4 2 3 5 4 4 6 0 1 6 4 6 4 6 3 4 4 4 6 6 1 6
 1 6 3 6 6 6 3 3 3 2 3 4 4 6 4 5 0 3 4 6 6 6 4 0 3 4 1 4 4 4 4 0 2 5 5 2 4
 6 2 4 4 4 6 6 6 4 6 4 6 2 6 3 0 4 4 6 6 3 2 6 6 4 6 1 4 4 4 3 0 4 4 2 6 4
 2 4 0 2 6 6 6 4 4 3 4 4 6 3 5 3 4 3 3 3 3 3 3 3 3 6 4 6 4 6 4 3 4 4 6 5 3
 4 5 2 4 0 4 6 2 0 6 6 6 4 6 6 3 6 5 5 3 1 5 4 6 3 6 3 3 2 6 3 3 6 3 0 4 6
 6 1 6 3 4 1 5 6 3 0 0 5 6 0 6 6 4 4 1 6 6 6 3 2 1 1 3 3 2 4 3 2 4 4 6 3 4
 6 3 6]
(336, 3)


Árbol de Desición
[2 2 6 6 4 2 6 1 0 4 2 1 4 6 4 1 6 5 4 5 3 1 6 1 6 4 0 3 6 6 3 4 5 6 4 0 4
 3 0 4 3 2 4 4 6 0 4 3 6 6 4 1 6 4 3 4 6 0 5 0 2 4 3 0 2 2 6 4 3 6 4 3 4 3
 3 1 3 6 6 6 5 6 4 0 6 0 5 5 4 4 4 4 2 3 3 6 6 6 6 6 6 4 1 5 4 4 6 4 0 2
 3 4 6 6 4 3 2 4 3 4 3 4 6 4 3 4 2 3 5 4 4 6 6 1 6 4 6 4 6 3 4 4 4 6 6 1 6
 1 6 3 6 6 6 3 3 3 2 3 4 4 6 4 5 2 3 4 6 6 6 4 0 3 4 1 4 4 4 4 0 2 5 5 2 4
 6 2 4 4 4 6 6 6 4 6 4 6 2 6 3 0 4 4 6 6 3 2 6 6 4 6 1 4 4 4 3 0 4 4 2 6 4
 2 4 0 0 6 6 0 4 4 3 4 4 6 3 5 3 4 3 3 3 3 3 3 3 3 6 4 6 4 6 4 3 4 4 6 5 3
 4 5 2 4 0 4 6 2 0 6 6 6 4 6 6 6 6 5 5 3 1 5 4 6 3 6 3 3 2 6 3 3 6 3 0 4 6
 6 1 6 3 4 1 5 6 3 2 0 5 6 0 6 6 4 4 1 6 6 6 3 2 1 1 3 3 2 4 3 2 4 4 6 3 4
 0 3 6]
(336, 21)
```

Biografía

https://claudiovz.github.io/scipy-lecture-notes-ES/intro/matplotlib/matplotlib.html

……