Este proyecto esta compuestro de los 3 sprints que se realizaron a lo largo del modulo, puede haber ligeras modificaciones de esto a lo que se llevo a cabo en cada sprints por los comentarios asi como los feedbacks recibidos

- Sprint 1: Desarrollar SVM y Softmax: En este caso vi el codigo que puso el profesor en los comentarios y se me hizo mas simple usar ese codigo hecho en clase debido al ahorro de transfromar las variables categoricas a numericas, en este caso solo se necesitan ajustar los parametros, es decir, de eliminan las categorias desde un principio y se configura de una manera mas rapida los modelos.
- sprint 2: Se entrenaron 4 arboles de decisiones pero añadi un 5 donde este no tiene ningun hiper parametro
- Sprint 3: Se desarrollo un ensemble asi como un random forest

Cabe resaltar que en cada sprint vuelvo a cargar la datra por las modificaciones que hago en cada uno de ellos

Librerias a usar

```
import numpy as np
import pandas as pd
import·seaborn·as·sns
import·matplotlib.pyplot·as·plt
from·google.colab·import·drive
from·sklearn.model_selection·import·train_test_split
from·sklearn.neighbors·import·KNeighborsClassifier
from·sklearn.metrics·import·accuracy_score
from·sklearn.linear_model·import·LogisticRegression
from·sklearn.pipeline·import·Pipeline
from·sklearn.preprocessing·import·StandardScaler
from sklearn.svm import LinearSVC
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn import tree
from sklearn.metrics import f1_score
from sklearn.ensemble import BaggingClassifier
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

sprint 1

Carga de datos

```
! pip install kaggle
! mkdir ~/.kaggle
! cp /content/drive/MyDrive/kaggle.json ~/.kaggle/kaggle.json
! chmod 600 ~/.kaggle/kaggle.json

! kaggle datasets download parulpandey/palmer-archipelago-antarctica-penguin-data
! unzip palmer-archipelago-antarctica-penguin-data.zip
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5.12)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from kagg
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from k
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from k
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (fron
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
mkdir: cannot create directory ‘/root/.kaggle’: File exists
palmer-archipelago-antarctica-penguin-data.zip: Skipping, found more recently modified I
Archive:  palmer-archipelago-antarctica-penguin-data.zip
replace penguins_lter.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: penguins_lter.csv
replace penguins_size.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: penguins_size.csv
```

Analisis de datos

```
penguins = pd.read_csv('penguins_size.csv').dropna()
penguins.head()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0

```
# Esta instrucción ayuda a comprobar si un DataFrame contiene valores nulos en cualquier atri
penguins.isnull().values.any()
```

```
False
```

Preparacion de datos

```
X = penguins.drop(["species","island","sex"], axis=1)

y = penguins.species.astype("category").cat.codes

X_train, X_test, y_train, y_test = train_test_split(
                        X, y, test_size=.2, random_state=42)


X_train, X_test, y_train, y_test = train_test_split(
                        X, y, test_size=.2, random_state=42)

X_train
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
230	40.9	13.7	214.0	4650.0
84	37.3	17.8	191.0	3350.0
303	50.0	15.9	224.0	5350.0
22	35.9	19.2	189.0	3800.0
29	40.5	18.9	180.0	3950.0
...	...	...	...	...
194	50.9	19.1	196.0	3550.0
77	37.2	19.4	184.0	3900.0
112	39.7	17.7	193.0	3200.0
277	45.5	15.0	220.0	5000.0
108	38.1	17.0	181.0	3175.0

267 rows × 4 columns

Modelos

SVM

```
svm_classifier_best = Pipeline([
                           ("scaler", StandardScaler()),
                           ("linear_svc", LinearSVC(C=2, loss="squared_hinge", fit_intercept=
])

svm_classifier_best.fit(X_train, y_train)
```

```
Pipeline(steps=[('scaler', StandardScaler()),
                ('linear_svc', LinearSVC(C=2, fit_intercept=False))])
```

```
y_pred = svm_classifier_best.predict(X_test)
accuracy_score(y_test, y_pred)
```

```
0.9850746268656716
```

```
param_grid = [
              {'C': [0.1, 0.5, 1, 2, 5], 'loss': ['hinge','squared_hinge']},
              {'C': [0.1, 0.5, 1, 2, 5], 'loss': ['hinge','squared_hinge'], 'fit_intercept':
]

svm_grid = LinearSVC(max_iter=10000)

grid_search = GridSearchCV(svm_grid, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True,
                          )

grid_search.fit(X_train, y_train)
```

```
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
```

```
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning:
  ConvergenceWarning,
GridSearchCV(cv=5, estimator=LinearSVC(max_iter=10000),
             param_grid=[{'C': [0.1, 0.5, 1, 2, 5],
                          'loss': ['hinge', 'squared_hinge']},
                         {'C': [0.1, 0.5, 1, 2, 5], 'fit_intercept': [False],
```

```
grid_search.best_estimator_
```

```
LinearSVC(C=5, fit_intercept=False, max_iter=10000)
```

```
cvresults = grid_search.cv_results_
for mean_score, params in zip(cvresults["mean_test_score"], cvresults["params"]):
  print(np.sqrt(-mean_score), params)
```

```
0.39298531311519613 {'C': 0.1, 'loss': 'hinge'}
0.38308031026015243 {'C': 0.1, 'loss': 'squared_hinge'}
0.6566330909059309 {'C': 0.5, 'loss': 'hinge'}
0.6753625731199567 {'C': 0.5, 'loss': 'squared_hinge'}
0.541305717505674 {'C': 1, 'loss': 'hinge'}
0.5752889469360648 {'C': 1, 'loss': 'squared_hinge'}
0.707748864799822 {'C': 2, 'loss': 'hinge'}
0.6809784550075622 {'C': 2, 'loss': 'squared_hinge'}
0.4985653772173597 {'C': 5, 'loss': 'hinge'}
0.610071538117657 {'C': 5, 'loss': 'squared_hinge'}
0.5464452177125045 {'C': 0.1, 'fit_intercept': False, 'loss': 'hinge'}
0.6039122394950113 {'C': 0.1, 'fit_intercept': False, 'loss': 'squared_hinge'}
0.4863618700343812 {'C': 0.5, 'fit_intercept': False, 'loss': 'hinge'}
0.7131587395809371 {'C': 0.5, 'fit_intercept': False, 'loss': 'squared_hinge'}
0.6799001036500849 {'C': 1, 'fit_intercept': False, 'loss': 'hinge'}
0.4251032677596799 {'C': 1, 'fit_intercept': False, 'loss': 'squared_hinge'}
0.47103377396152696 {'C': 2, 'fit_intercept': False, 'loss': 'hinge'}
0.6199569426675574 {'C': 2, 'fit_intercept': False, 'loss': 'squared_hinge'}
0.5450366863962642 {'C': 5, 'fit_intercept': False, 'loss': 'hinge'}
0.3578997669551949 {'C': 5, 'fit_intercept': False, 'loss': 'squared_hinge'}
```

```
svm_classifier_best = Pipeline([
                           ("scaler", StandardScaler()),
                           ("linear_svc", LinearSVC(C=5, loss="hinge", fit_intercept=False, m
])

svm_classifier_best.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning: Li
  ConvergenceWarning,
Pipeline(steps=[('scaler', StandardScaler()),
                ('linear_svc',
                 LinearSVC(C=5, fit_intercept=False, loss='hinge',
                           max_iter=10000))])
```

```
y_pred = svm_classifier_best.predict(X_test)
accuracy_score(y_test, y_pred)
```

```
0.9850746268656716
```

Softmax

```
softmax_reg = LogisticRegression(multi_class="multinomial", solver="lbfgs", max_iter=10000)
softmax_reg.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=10000, multi_class='multinomial')
```

```
score = softmax_reg.score(X, y)
score
```

```
0.9970059880239521
```

Splint 2

carga de datos

```
! cp /content/drive/MyDrive/kaggle.json ~/.kaggle/kaggle.json
! chmod 600 ~/.kaggle/kaggle.json

! kaggle datasets download parulpandey/palmer-archipelago-antarctica-penguin-data
! unzip palmer-archipelago-antarctica-penguin-data.zip
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5.12)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from kagg
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages 
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from k
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from k
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from 
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packag
mkdir: cannot create directory ‘/root/.kaggle’: File exists
palmer-archipelago-antarctica-penguin-data.zip: Skipping, found more recently modified l
Archive:  palmer-archipelago-antarctica-penguin-data.zip
replace penguins_lter.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: penguins_lter.csv
replace penguins_size.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: penguins_size.csv
```

Analisis de datos

```
penguins = pd.read_csv('penguins_size.csv').dropna()
penguins.head()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0

```
# Esta instrucción ayuda a comprobar si un DataFrame contiene valores nulos en cualquier atri
penguins.isnull().values.any()
```

```
False
```

Preparacion de datos

```
X = penguins.drop(["species","island","sex"], axis=1)

y = penguins.species.astype("category").cat.codes

# X_train, X_test, y_train, y_test = train_test_split(
#                          X, y, test_size=.2, random_state=42)


X_train, X_test, y_train, y_test = train_test_split(
                        X, y, test_size=.4, random_state=42)

X_train
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
28	37.9	18.6	172.0	3150.0
42	36.0	18.5	186.0	3100.0
253	59.6	17.0	230.0	6050.0
73	45.8	18.9	197.0	4150.0
273	50.1	15.0	225.0	5000.0
...	...	...	...	...
194	50.9	19.1	196.0	3550.0
77	37.2	19.4	184.0	3900.0
112	39.7	17.7	193.0	3200.0
277	45.5	15.0	220.0	5000.0
108	38.1	17.0	181.0	3175.0

200 rows × 4 columns

```
X_nombres = list(X.columns)
especies = penguins.species.unique()
```

Arboles

```
tree1 = DecisionTreeClassifier(max_depth=4)
tree1.fit(X_train, y_train)
tree2 = DecisionTreeClassifier(max_depth=4, min_samples_split=8)
tree2.fit(X_train, y_train)
tree3 = DecisionTreeClassifier(max_depth=4, min_samples_split=8, max_features=3)
tree3.fit(X_train, y_train)
tree4 = DecisionTreeClassifier(min_samples_split=8, max_features=3)
tree4.fit(X_train, y_train)
tree5 = DecisionTreeClassifier()
tree5.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```
print("Predicciones sobre Test")
y_prediction1 = tree1.predict(X_test)
print("Arbol 1 " +str(accuracy_score(y_test, y_prediction1)))

y_prediction2 = tree2.predict(X_test)
print("Arbol 2 " +str(accuracy_score(y_test, y_prediction2)))

y_prediction3 = tree3.predict(X_test)
print("Arbol 3 " +str (accuracy_score(y_test, y_prediction3)))

y_prediction4 = tree4.predict(X_test)
print("Arbol 4 " +str(accuracy_score(y_test, y_prediction4)))

y_prediction5 = tree5.predict(X_test)
print("Arbol 5 " +str(accuracy_score(y_test, y_prediction5)))
```

```
Predicciones sobre Test
Arbol 1 0.9701492537313433
Arbol 2 0.9552238805970149
Arbol 3 0.9701492537313433
Arbol 4 0.9626865671641791
Arbol 5 0.9626865671641791
```

```
print("Predicciones sobre Trining")
y_prediction1t = tree1.predict(X_train)
print("Arbol 1 " +str(accuracy_score(y_train, y_prediction1t)))

y_prediction2t = tree2.predict(X_train)
print("Arbol 2 " +str(accuracy_score(y_train, y_prediction2t)))
```

```
y_prediction3t = tree3.predict(X_train)
print("Arbol 3 " +str (accuracy_score(y_train, y_prediction3t)))

y_prediction4t = tree4.predict(X_train)
print("Arbol 4 " +str(accuracy_score(y_train, y_prediction4t)))

y_prediction5t = tree5.predict(X_train)
print("Arbol 5 " +str(accuracy_score(y_train, y_prediction5t)))
```

```
Predicciones sobre Trining
Arbol 1 0.985
Arbol 2 0.985
Arbol 3 0.985
Arbol 4 0.995
Arbol 5 1.0
```

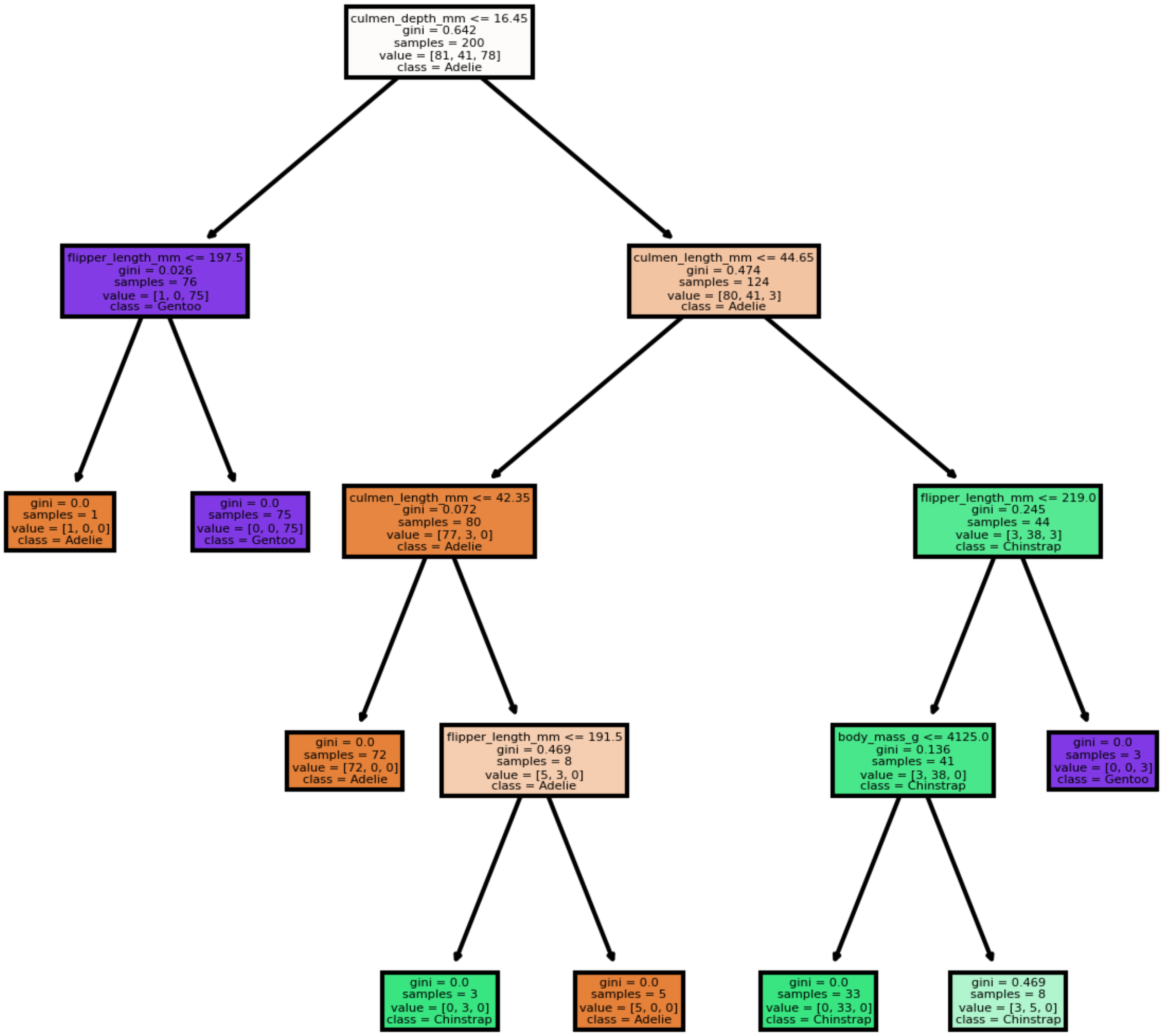
```
y_prediction1
```

```
array([0, 2, 0, 1, 0, 2, 2, 1, 1, 1, 0, 0, 2, 0, 2, 0, 0, 1, 0, 2, 0, 0,
       2, 1, 0, 0, 2, 2, 1, 2, 1, 2, 0, 0, 2, 2, 1, 2, 0, 0, 0, 0, 1, 1,
       0, 0, 2, 0, 0, 2, 0, 2, 2, 0, 0, 2, 0, 0, 1, 1, 2, 2, 2, 0, 0, 2,
       0, 2, 0, 2, 0, 0, 0, 2, 2, 1, 0, 0, 0, 2, 0, 0, 0, 0, 2, 2, 0, 2,
       1, 1, 2, 2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 2, 0, 1, 0, 2, 2, 0, 1,
       0, 2, 0, 1, 1, 1, 0, 2, 0, 0, 0, 2, 2, 0, 0, 2, 0, 0, 0, 1, 0, 0,
       1, 0], dtype=int8)
```

```
print(confusion_matrix(y_test,y_prediction1))
print(classification_report(y_test,y_prediction1))
```

```
[[65  0  0]
 [ 2 24  1]
 [ 1  0 41]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        65
           1       1.00      0.89      0.94        27
           2       0.98      0.98      0.98        42

    accuracy                           0.97       134
   macro avg       0.98      0.96      0.96       134
weighted avg       0.97      0.97      0.97       134
```

```
fig, acex = plt.subplots(nrows =1 ,ncols =1 ,figsize=(5,5), dpi=300)
tree.plot_tree(tree1,
               feature_names = X_nombres,
               class_names = especies,
               filled = True)
fig.show()
```



culmen_depth_mm <= 16.45
gini = 0.642
samples = 200
value = [81, 41, 78]
class = Adelie
flipper_length_mm <= 197.5
gini = 0.026
samples = 76
value = [1, 0, 75]
class = Gentoo
culmen_length_mm <= 44.65
gini = 0.474
samples = 124
value = [80, 41, 3]
class = Adelie
gini = 0.0
samples = 1
value = [1, 0, 0]
class = Adelie
gini = 0.0
samples = 75
value = [0, 0, 75]
class = Gentoo
culmen_length_mm <= 42.35
gini = 0.072
samples = 80
value = [77, 3, 0]
class = Adelie
flipper_length_mm <= 219.0
gini = 0.245
samples = 44
value = [3, 38, 3]
class = Chinstrap
gini = 0.0
samples = 72
value = [72, 0, 0]
class = Adelie
flipper_length_mm <= 191.5
gini = 0.469
samples = 8
value = [5, 3, 0]
class = Adelie
body_mass_g <= 4125.0
gini = 0.136
samples = 41
value = [3, 38, 0]
class = Chinstrap
gini = 0.0
samples = 3
value = [0, 0, 3]
class = Gentoo
gini = 0.0
samples = 3
value = [0, 3, 0]
class = Chinstrap
gini = 0.0
samples = 5
value = [5, 0, 0]
class = Adelie
gini = 0.0
samples = 33
value = [0, 33, 0]
class = Chinstrap
gini = 0.469
samples = 8
value = [3, 5, 0]
class = Chinstrap

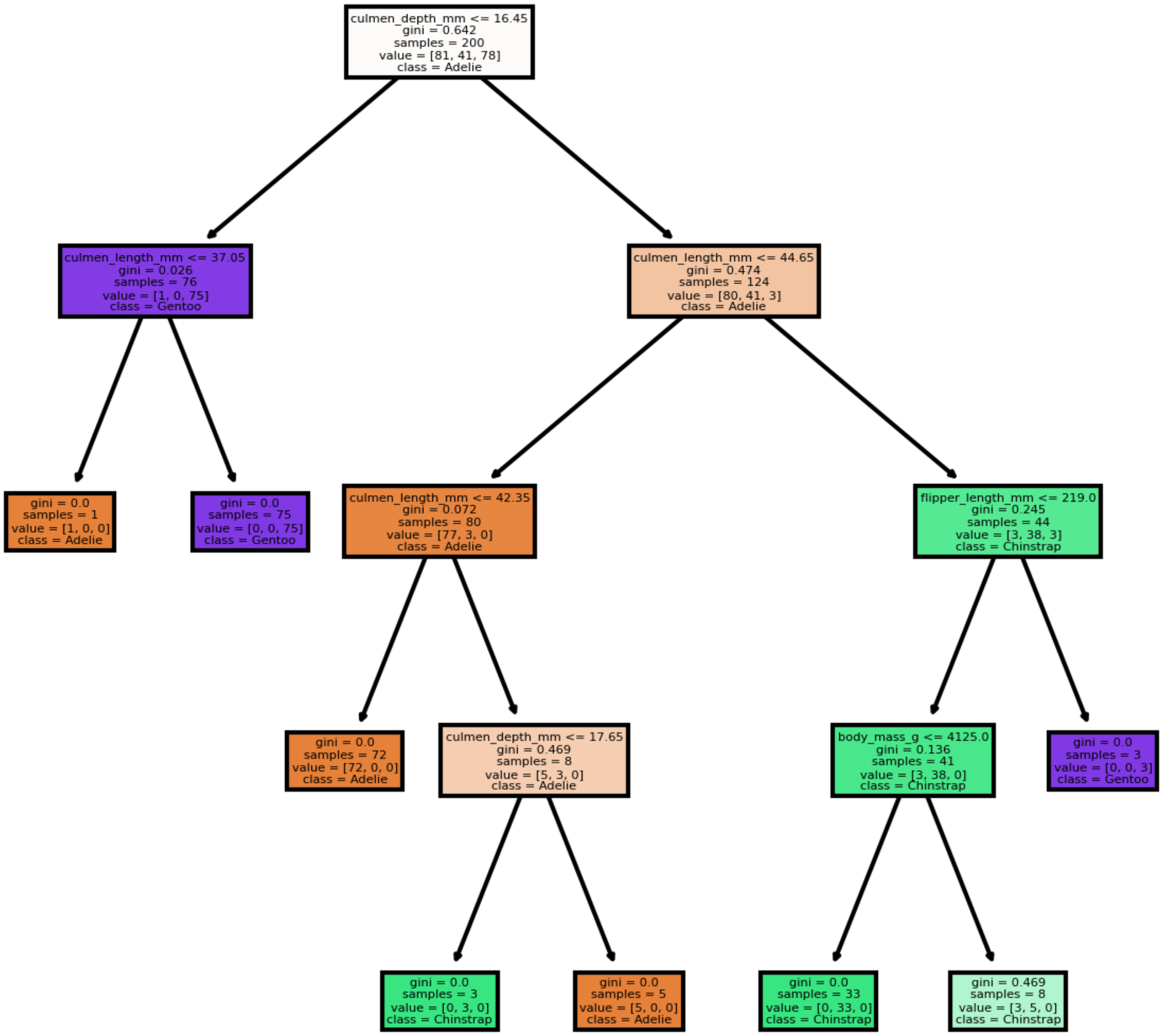
```
y_prediction2
```

```
array([0, 2, 0, 1, 0, 2, 2, 1, 1, 1, 1, 0, 2, 0, 2, 0, 0, 1, 0, 2, 0, 0,
       2, 1, 0, 0, 2, 2, 1, 2, 1, 2, 0, 0, 2, 2, 1, 2, 0, 0, 0, 0, 1, 1,
       0, 0, 2, 0, 0, 2, 0, 2, 2, 0, 0, 2, 0, 0, 1, 1, 2, 2, 2, 0, 0, 2,
       0, 2, 0, 2, 0, 0, 1, 2, 2, 1, 0, 0, 0, 2, 0, 0, 0, 0, 2, 2, 0, 2,
       1, 1, 2, 2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 2, 0, 1, 0, 2, 2, 0, 1,
       0, 2, 0, 1, 1, 1, 0, 2, 0, 0, 0, 2, 2, 0, 0, 2, 0, 0, 0, 1, 2, 0,
       1, 0], dtype=int8)
```

```
print(confusion_matrix(y_test,y_prediction2))
print(classification_report(y_test,y_prediction2))
```

```
[[63  1  1]
 [ 2 24  1]
 [ 0  1 41]]
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        65
           1       0.92      0.89      0.91        27
           2       0.95      0.98      0.96        42

    accuracy                           0.96       134
   macro avg       0.95      0.94      0.95       134
weighted avg       0.95      0.96      0.96       134
```

```
fig, acex = plt.subplots(nrows =1 ,ncols =1 ,figsize=(5,5), dpi=300)
tree.plot_tree(tree2,
               feature_names = X_nombres,
               class_names = especies,
               filled = True)
fig.show()
```


culmen_depth_mm <= 16.45
gini = 0.642
samples = 200
value = [81, 41, 78]
class = Adelie
culmen_length_mm <= 37.05
gini = 0.026
samples = 76
value = [1, 0, 75]
class = Gentoo
culmen_length_mm <= 44.65
gini = 0.474
samples = 124
value = [80, 41, 3]
class = Adelie
gini = 0.0
samples = 1
value = [1, 0, 0]
class = Adelie
gini = 0.0
samples = 75
value = [0, 0, 75]
class = Gentoo
culmen_length_mm <= 42.35
gini = 0.072
samples = 80
value = [77, 3, 0]
class = Adelie
flipper_length_mm <= 219.0
gini = 0.245
samples = 44
value = [3, 38, 3]
class = Chinstrap
gini = 0.0
samples = 72
value = [72, 0, 0]
class = Adelie
culmen_depth_mm <= 17.65
gini = 0.469
samples = 8
value = [5, 3, 0]
class = Adelie
body_mass_g <= 4125.0
gini = 0.136
samples = 41
value = [3, 38, 0]
class = Chinstrap
gini = 0.0
samples = 3
value = [0, 0, 3]
class = Gentoo
gini = 0.0
samples = 3
value = [0, 3, 0]
class = Chinstrap
gini = 0.0
samples = 5
value = [5, 0, 0]
class = Adelie
gini = 0.0
samples = 33
value = [0, 33, 0]
class = Chinstrap
gini = 0.469
samples = 8
value = [3, 5, 0]
class = Chinstrap

```
y_prediction3
```

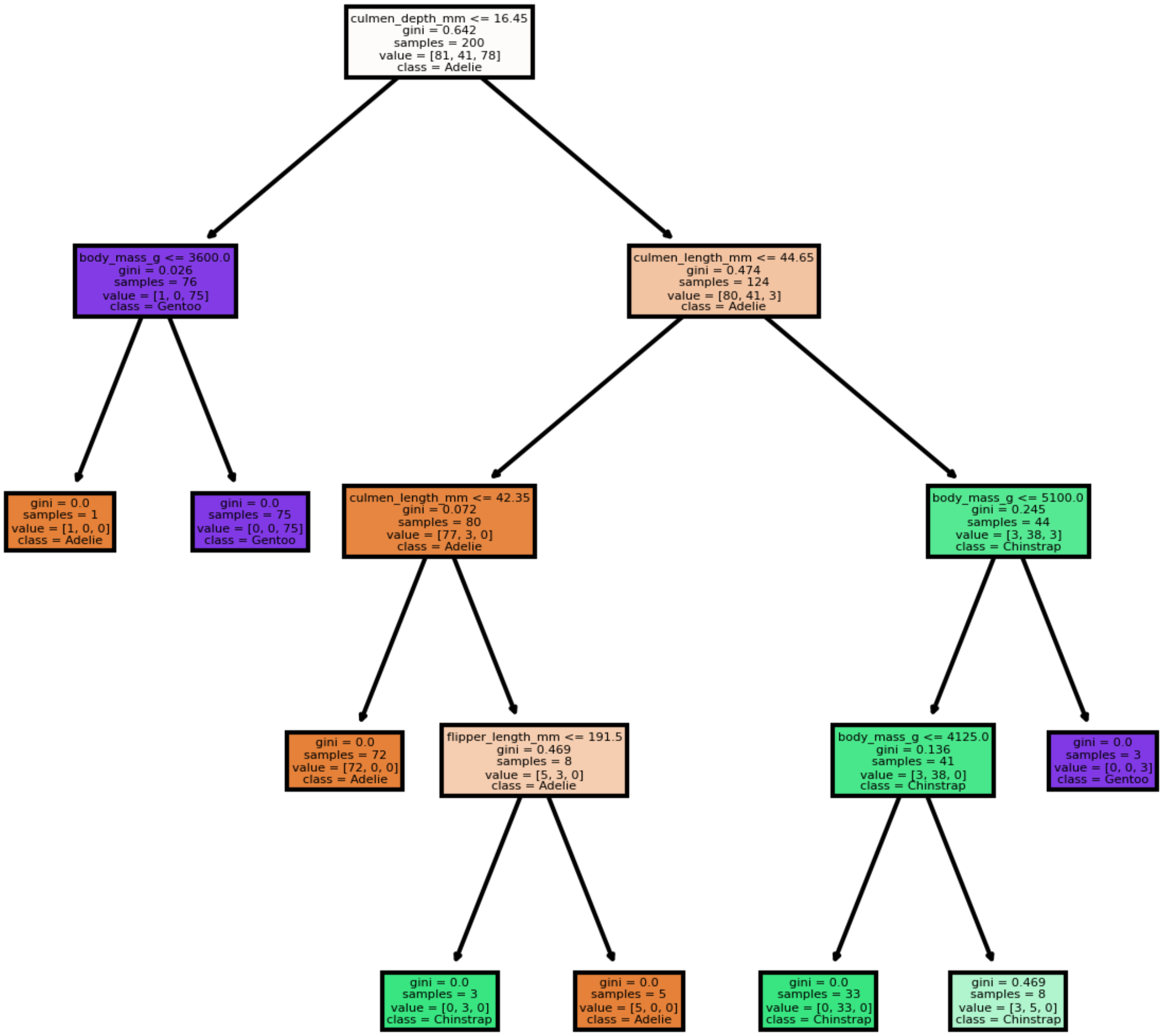
```
array([0, 2, 0, 1, 0, 2, 2, 1, 1, 1, 0, 0, 2, 0, 2, 0, 0, 1, 0, 2, 0, 0,
       2, 1, 0, 0, 2, 2, 1, 2, 1, 2, 0, 0, 2, 2, 1, 2, 0, 0, 0, 0, 1, 1,
       0, 0, 2, 0, 0, 2, 0, 2, 2, 0, 0, 2, 0, 0, 1, 1, 2, 2, 2, 0, 0, 2,
       0, 2, 0, 2, 0, 0, 0, 2, 2, 1, 0, 0, 0, 2, 0, 0, 0, 0, 2, 2, 0, 2,
       1, 1, 2, 2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 2, 0, 1, 0, 2, 2, 0, 1,
       0, 2, 0, 1, 1, 1, 0, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 1, 0, 0,
       1, 0], dtype=int8)
```

```
# tree.plot_tree(tree3)
```

```
print(confusion_matrix(y_test,y_prediction3))
print(classification_report(y_test,y_prediction3))
```

```
[[65  0  0]
 [ 3 24  0]
 [ 1  0 41]]
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        65
           1       1.00      0.89      0.94        27
           2       1.00      0.98      0.99        42

    accuracy                           0.97       134
   macro avg       0.98      0.96      0.97       134
weighted avg       0.97      0.97      0.97       134
```

```
fig, acex = plt.subplots(nrows =1 ,ncols =1 ,figsize=(5,5), dpi=300)
tree.plot_tree(tree3,
               feature_names = X_nombres,
               class_names = especies,
               filled = True)
fig.show()
```


culmen_depth_mm <= 16.45
gini = 0.642
samples = 200
value = [81, 41, 78]
class = Adelie
body_mass_g <= 3600.0
gini = 0.026
samples = 76
value = [1, 0, 75]
class = Gentoo
culmen_length_mm <= 44.65
gini = 0.474
samples = 124
value = [80, 41, 3]
class = Adelie
gini = 0.0
samples = 1
value = [1, 0, 0]
class = Adelie
gini = 0.0
samples = 75
value = [0, 0, 75]
class = Gentoo
culmen_length_mm <= 42.35
gini = 0.072
samples = 80
value = [77, 3, 0]
class = Adelie
body_mass_g <= 5100.0
gini = 0.245
samples = 44
value = [3, 38, 3]
class = Chinstrap
gini = 0.0
samples = 72
value = [72, 0, 0]
class = Adelie
flipper_length_mm <= 191.5
gini = 0.469
samples = 8
value = [5, 3, 0]
class = Adelie
body_mass_g <= 4125.0
gini = 0.136
samples = 41
value = [3, 38, 0]
class = Chinstrap
gini = 0.0
samples = 3
value = [0, 0, 3]
class = Gentoo
gini = 0.0
samples = 3
value = [0, 3, 0]
class = Chinstrap
gini = 0.0
samples = 5
value = [5, 0, 0]
class = Adelie
gini = 0.0
samples = 33
value = [0, 33, 0]
class = Chinstrap
gini = 0.469
samples = 8
value = [3, 5, 0]
class = Chinstrap

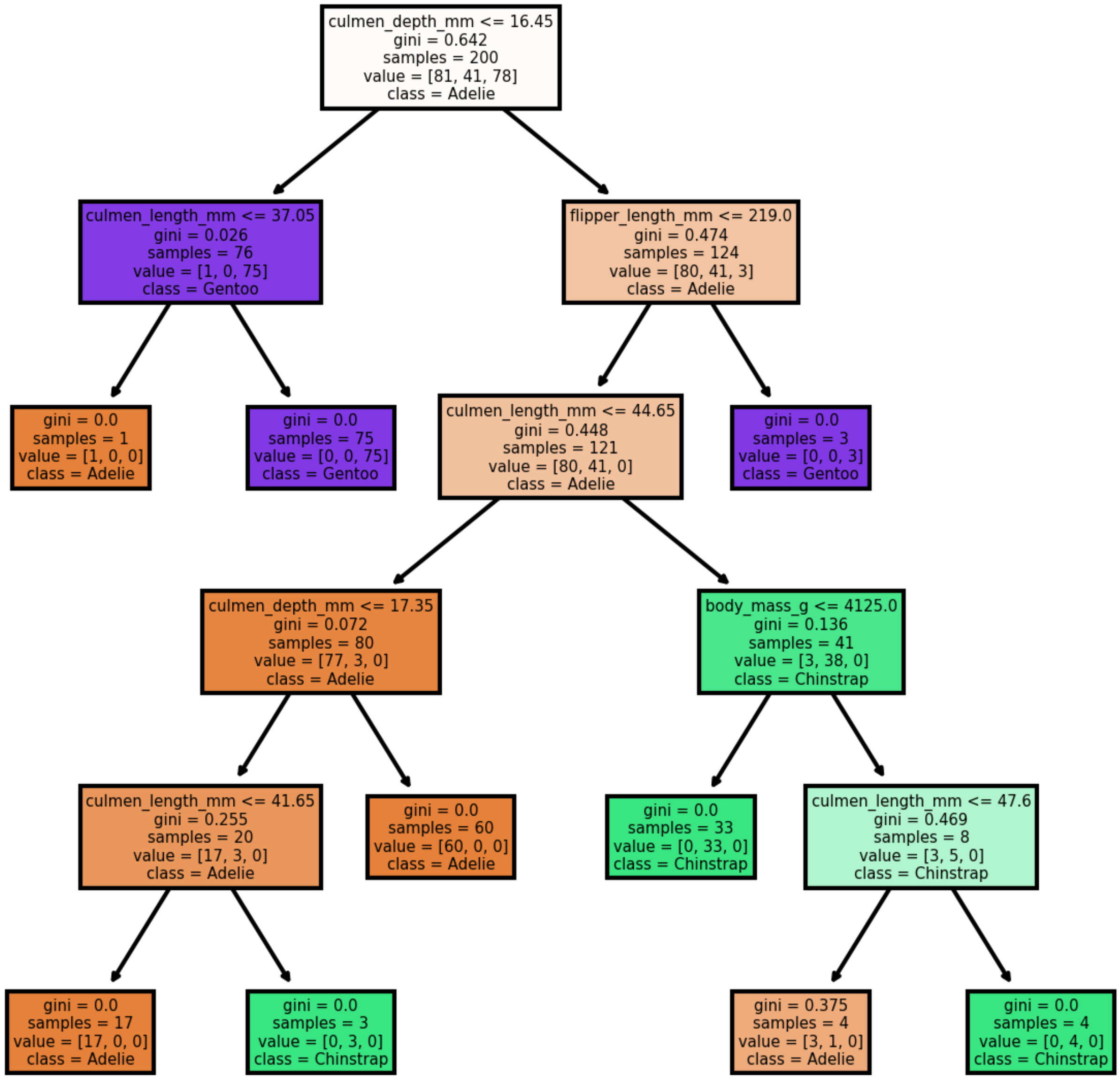
```
y_prediction4
```

```
array([0, 2, 0, 1, 0, 2, 2, 1, 1, 1, 0, 0, 2, 0, 2, 0, 0, 1, 0, 2, 0, 0,
       2, 1, 0, 0, 2, 2, 1, 2, 1, 2, 0, 0, 2, 2, 1, 2, 0, 0, 0, 0, 1, 1,
       0, 0, 2, 0, 0, 2, 0, 2, 2, 0, 0, 2, 0, 0, 1, 1, 2, 2, 2, 0, 0, 2,
       0, 2, 0, 2, 0, 0, 1, 2, 2, 1, 0, 0, 0, 2, 0, 0, 0, 0, 2, 2, 0, 2,
       1, 1, 2, 2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 2, 0, 1, 0, 2, 2, 0, 1,
       0, 2, 0, 1, 1, 1, 0, 2, 0, 0, 0, 2, 2, 0, 0, 2, 0, 0, 0, 1, 2, 0,
       1, 0], dtype=int8)
```

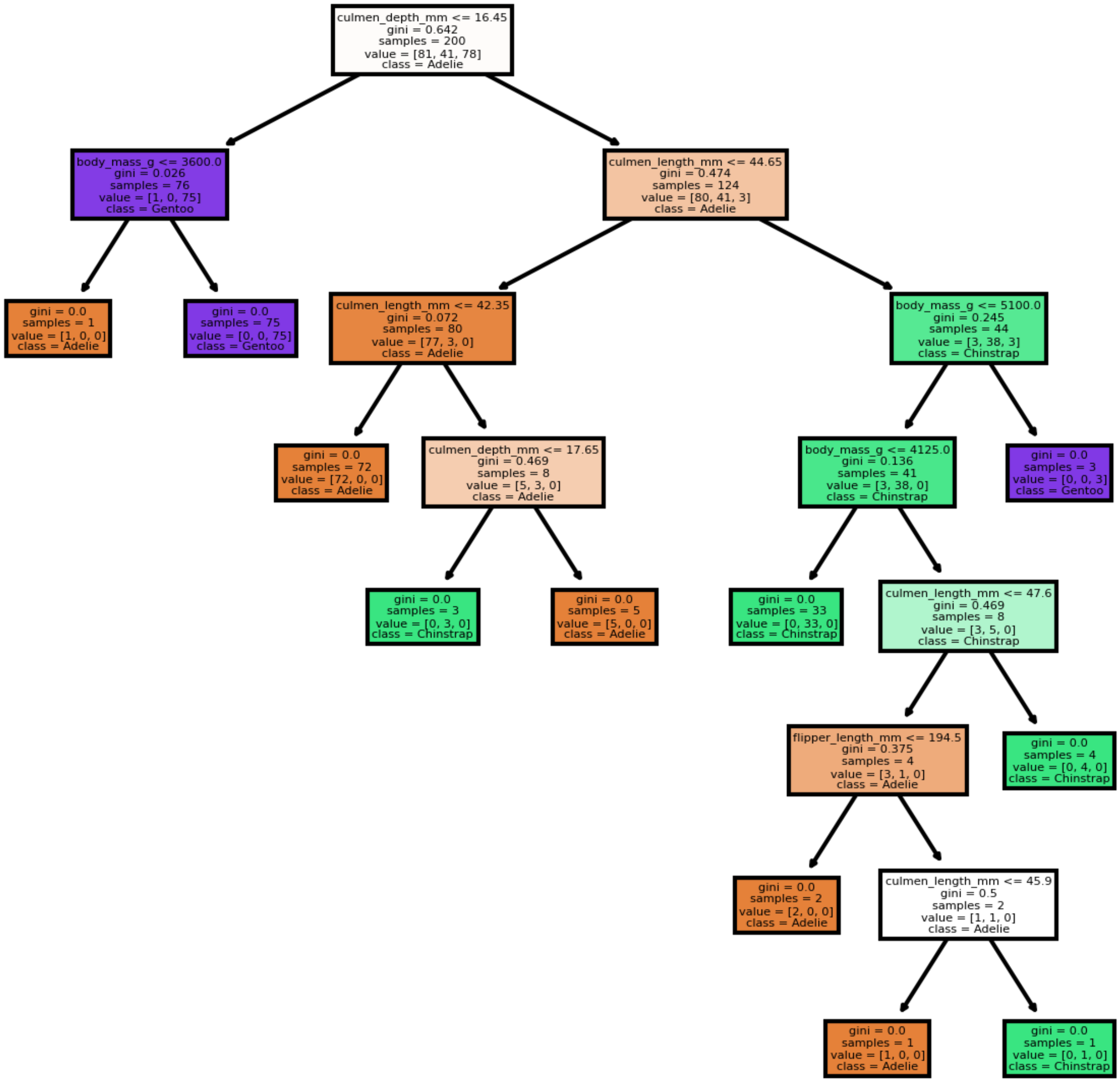
```
print(confusion_matrix(y_test,y_prediction4))
print(classification_report(y_test,y_prediction4))
```

```
[[64  0  1]
 [ 2 24  1]
 [ 0  1 41]]
              precision    recall  f1-score   support

           0       0.97      0.98      0.98        65
           1       0.96      0.89      0.92        27
           2       0.95      0.98      0.96        42

    accuracy                           0.96       134
   macro avg       0.96      0.95      0.95       134
weighted avg       0.96      0.96      0.96       134
```

```
fig, acex = plt.subplots(nrows =1 ,ncols =1 ,figsize=(5,5), dpi=300)
tree.plot_tree(tree4,
               feature_names = X_nombres,
               class_names = especies,
               filled = True)
fig.show()
```


culmen_depth_mm <= 16.45
gini = 0.642
samples = 200
value = [81, 41, 78]
class = Adelie
culmen_length_mm <= 37.05
gini = 0.026
samples = 76
value = [1, 0, 75]
class = Gentoo
flipper_length_mm <= 219.0
gini = 0.474
samples = 124
value = [80, 41, 3]
class = Adelie
gini = 0.0
samples = 1
value = [1, 0, 0]
class = Adelie
gini = 0.0
samples = 75
value = [0, 0, 75]
class = Gentoo
culmen_length_mm <= 44.65
gini = 0.448
samples = 121
value = [80, 41, 0]
class = Adelie
gini = 0.0
samples = 3
value = [0, 0, 3]
class = Gentoo
culmen_depth_mm <= 17.35
gini = 0.072
samples = 80
value = [77, 3, 0]
class = Adelie
body_mass_g <= 4125.0
gini = 0.136
samples = 41
value = [3, 38, 0]
class = Chinstrap
culmen_length_mm <= 41.65
gini = 0.255
samples = 20
value = [17, 3, 0]
class = Adelie
gini = 0.0
samples = 60
value = [60, 0, 0]
class = Adelie
gini = 0.0
samples = 33
value = [0, 33, 0]
class = Chinstrap
culmen_length_mm <= 47.6
gini = 0.469
samples = 8
value = [3, 5, 0]
class = Chinstrap
gini = 0.0
samples = 17
value = [17, 0, 0]
class = Adelie
gini = 0.0
samples = 3
value = [0, 3, 0]
class = Chinstrap
gini = 0.375
samples = 4
value = [3, 1, 0]
class = Adelie
gini = 0.0
samples = 4
value = [0, 4, 0]
class = Chinstrap

```
y_prediction5

     array([0, 2, 0, 1, 0, 2, 2, 1, 1, 1, 1, 0, 2, 0, 2, 0, 0, 1, 0, 2, 0, 0,
            2, 1, 0, 0, 2, 2, 1, 2, 1, 2, 0, 0, 2, 2, 1, 2, 0, 0, 0, 0, 1, 1,
            0, 0, 2, 0, 0, 2, 0, 2, 2, 0, 0, 2, 0, 0, 1, 1, 2, 2, 2, 0, 0, 2,
            0, 2, 0, 2, 0, 0, 1, 2, 2, 1, 0, 0, 0, 2, 0, 0, 0, 0, 2, 2, 0, 2,
            1, 1, 2, 2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 2, 0, 1, 0, 2, 2, 0, 1,
            0, 2, 0, 1, 1, 1, 0, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 1, 0, 0,
            1, 0], dtype=int8)


print(confusion_matrix(y_test,y_prediction5))
print(classification_report(y_test,y_prediction5))

     [[64  1  0]
      [ 3 24  0]
      [ 0  1 41]]
                   precision    recall  f1-score   support

                0       0.96      0.98      0.97        65
                1       0.92      0.89      0.91        27
                2       1.00      0.98      0.99        42

         accuracy                           0.96       134
        macro avg       0.96      0.95      0.95       134
     weighted avg       0.96      0.96      0.96       134



fig, acex = plt.subplots(nrows =1 ,ncols =1 ,figsize=(5,5), dpi=300)
tree.plot_tree(tree5,
               feature_names = X_nombres,
               class_names = especies,
               filled = True)
fig.show()
```


culmen_depth_mm <= 16.45
gini = 0.642
samples = 200
value = [81, 41, 78]
class = Adelie
body_mass_g <= 3600.0
gini = 0.026
samples = 76
value = [1, 0, 75]
class = Gentoo
culmen_length_mm <= 44.65
gini = 0.474
samples = 124
value = [80, 41, 3]
class = Adelie
gini = 0.0
samples = 1
value = [1, 0, 0]
class = Adelie
gini = 0.0
samples = 75
value = [0, 0, 75]
class = Gentoo
culmen_length_mm <= 42.35
gini = 0.072
samples = 80
value = [77, 3, 0]
class = Adelie
body_mass_g <= 5100.0
gini = 0.245
samples = 44
value = [3, 38, 3]
class = Chinstrap
gini = 0.0
samples = 72
value = [72, 0, 0]
class = Adelie
culmen_depth_mm <= 17.65
gini = 0.469
samples = 8
value = [5, 3, 0]
class = Adelie
body_mass_g <= 4125.0
gini = 0.136
samples = 41
value = [3, 38, 0]
class = Chinstrap
gini = 0.0
samples = 3
value = [0, 0, 3]
class = Gentoo
gini = 0.0
samples = 3
value = [0, 3, 0]
class = Chinstrap
gini = 0.0
samples = 5
value = [5, 0, 0]
class = Adelie
gini = 0.0
samples = 33
value = [0, 33, 0]
class = Chinstrap
culmen_length_mm <= 47.6
gini = 0.469
samples = 8
value = [3, 5, 0]
class = Chinstrap
flipper_length_mm <= 194.5
gini = 0.375
samples = 4
value = [3, 1, 0]
class = Adelie
gini = 0.0
samples = 4
value = [0, 4, 0]
class = Chinstrap
gini = 0.0
samples = 2
value = [2, 0, 0]
class = Adelie
culmen_length_mm <= 45.9
gini = 0.5
samples = 2
value = [1, 1, 0]
class = Adelie
gini = 0.0
samples = 1
value = [1, 0, 0]
class = Adelie
gini = 0.0
samples = 1
value = [0, 1, 0]
class = Chinstrap

Sprint 3

Craga de datos

```
! cp /content/drive/MyDrive/kaggle.json ~/.kaggle/kaggle.json
! chmod 600 ~/.kaggle/kaggle.json
! kaggle datasets download parulpandey/palmer-archipelago-antarctica-penguin-data
! unzip palmer-archipelago-antarctica-penguin-data.zip
```

```
palmer-archipelago-antarctica-penguin-data.zip: Skipping, found more recently modified l
Archive:  palmer-archipelago-antarctica-penguin-data.zip
replace penguins_lter.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: penguins_lter.csv
replace penguins_size.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: penguins_size.csv
```

Analisis de datos

```
import pandas as pd
penguins = pd.read_csv('penguins_size.csv')
penguins.head()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
3	Adelie	Torgersen	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0

```
penguins.describe()
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
count	342.000000	342.000000	342.000000	342.000000

```
penguins.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    object 
 1   island             344 non-null    object 
 2   culmen_length_mm   342 non-null    float64
 3   culmen_depth_mm    342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                334 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

Transformacion de datos

```
penguins.drop('sex', axis=1,inplace=True)
df_gender = pd.get_dummies(penguins['island'])
penguins.drop('island', axis=1,inplace=True)
penguins = pd.concat([penguins, df_gender], axis=1)
penguins
```

	species	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	Biscoe
343	Gentoo	49.9	16.1	213.0	5400.0	1

```
penguins.isna().any()
```

```
species              False
culmen_length_mm      True
culmen_depth_mm       True
flipper_length_mm     True
body_mass_g           True
Biscoe               False
Dream                False
Torgersen            False
dtype: bool
```

```
penguins['culmen_length_mm'].fillna((penguins['culmen_length_mm'].mean()), inplace=True)
penguins['culmen_depth_mm'].fillna((penguins['culmen_depth_mm'].mean()), inplace=True)
penguins['flipper_length_mm'].fillna((penguins['flipper_length_mm'].mean()), inplace=True)
penguins['body_mass_g'].fillna((penguins['body_mass_g'].mean()), inplace=True)
```

```
from sklearn.model_selection import train_test_split

X = penguins.drop("species", axis=1)

y = penguins.species.astype("category").cat.codes
# Usar esta instrucción cuando se requieran etiquetas de clase codificadas numéricamente, p.e
# O si vamos a hacer búsqueda de hiperparámetros para optimizar modelos.


X_train, X_test, y_train, y_test = train_test_split(
                        X, y, test_size=.2, random_state=42)
```

transformacion de categoricas

```
X_train_num = X_train.drop(["Biscoe","Dream","Torgersen"], axis=1) # Obtener una versión solo
from sklearn.impute import SimpleImputer
num_imputer = SimpleImputer(strategy="median") # Rellenar valores perdidos de atributos numér
X_train_num_array = num_imputer.fit_transform(X_train_num)
X_train_num = pd.DataFrame(X_train_num_array, columns=X_train_num.columns, index=X_train_num.
X_train_num.head()
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
66	35.5	16.2	195.0	3350.0

```
from sklearn.impute import SimpleImputer

num_imputer = SimpleImputer(strategy="median") # Rellenar valores perdidos de atributos numér

X_train_num_array = num_imputer.fit_transform(X_train_num)
X_train_num = pd.DataFrame(X_train_num_array, columns=X_train_num.columns, index=X_train_num.
X_train_num.head()
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
66	35.5	16.2	195.0	3350.0
229	46.8	15.4	215.0	5150.0
7	39.2	19.6	195.0	4675.0
140	40.2	17.1	193.0	3400.0
323	49.1	15.0	228.0	5500.0

```
X_train_num = X_train.drop(["Biscoe","Dream","Torgersen"], axis=1) # Obtener una versión solo
from sklearn.impute import SimpleImputer
num_imputer = SimpleImputer(strategy="median") # Rellenar valores perdidos de atributos numér
X_train_num_array = num_imputer.fit_transform(X_train_num)
X_train_num = pd.DataFrame(X_train_num_array, columns=X_train_num.columns, index=X_train_num.
X_train_num.head()
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
66	35.5	16.2	195.0	3350.0
229	46.8	15.4	215.0	5150.0
7	39.2	19.6	195.0	4675.0
140	40.2	17.1	193.0	3400.0
323	49.1	15.0	228.0	5500.0

```
from sklearn.impute import SimpleImputer

num_imputer = SimpleImputer(strategy="median") # Rellenar valores perdidos de atributos numér

X_train_num_array = num_imputer.fit_transform(X_train_num)
X_train_num = pd.DataFrame(X_train_num_array, columns=X_train_num.columns, index=X_train_num.
X_train_num.head()
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
66	35.5	16.2	195.0	3350.0
229	46.8	15.4	215.0	5150.0
7	39.2	19.6	195.0	4675.0
140	40.2	17.1	193.0	3400.0
323	49.1	15.0	228.0	5500.0

division de conjunto

```
X = penguins.drop('species', axis=1)
X
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	Biscoe	Dream
0	39.10000	18.70000	181.000000	3750.000000	0	0
1	39.50000	17.40000	186.000000	3800.000000	0	0
2	40.30000	18.00000	195.000000	3250.000000	0	0
3	43.92193	17.15117	200.915205	4201.754386	0	0
4	36.70000	19.30000	193.000000	3450.000000	0	0
...	...	...	...	...	...	...
339	43.92193	17.15117	200.915205	4201.754386	1	0
340	46.80000	14.30000	215.000000	4850.000000	1	0
341	50.40000	15.70000	222.000000	5750.000000	1	0
342	45.20000	14.80000	212.000000	5200.000000	1	0
343	49.90000	16.10000	213.000000	5400.000000	1	0

344 rows × 7 columns

```
y = penguins.species
y
```

```
0      Adelie
1      Adelie
2      Adelie
3      Adelie
4      Adelie
        ...
339    Gentoo
340    Gentoo
```

```
341    Gentoo
342    Gentoo
343    Gentoo
Name: species, Length: 344, dtype: object
```

4. Random Forests

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.4, random_state=42)

scaler = StandardScaler()

scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
# Las dos anteriores instrucciones serían equivalentes a:
# X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)
```

```
bagging_clf = BaggingClassifier(
    DecisionTreeClassifier(), n_estimators=200,
    max_samples=100, bootstrap=True, n_jobs=-1)

bagging_clf.fit(X_train, y_train)

y_pred10 = bagging_clf.predict(X_test[:10])
y_pred10
```

```
array(['Chinstrap', 'Chinstrap', 'Gentoo', 'Chinstrap', 'Gentoo',
       'Gentoo', 'Gentoo', 'Gentoo', 'Adelie', 'Gentoo'], dtype=object)
```

```
y_test[:10]
```

```
194    Chinstrap
157    Chinstrap
225       Gentoo
208    Chinstrap
318       Gentoo
329       Gentoo
319       Gentoo
260       Gentoo
114       Adelie
220       Gentoo
Name: species, dtype: object
```

```
y_pred = bagging_clf.predict(X_test)
```

```
print(classification_report(y_pred, y_test)) # Una visión más detallada del rendimiento por c
```

```
              precision    recall  f1-score   support

      Adelie       0.95      0.98      0.97        62
   Chinstrap       0.96      0.92      0.94        25
      Gentoo       1.00      0.98      0.99        51

    accuracy                           0.97       138
   macro avg       0.97      0.96      0.97       138
weighted avg       0.97      0.97      0.97       138
```

```
confusion_matrix(y_pred, y_test)
```

```
array([[61,  1,  0],
       [ 2, 23,  0],
       [ 1,  0, 50]])
```

```
accuracy_score(y_pred, y_test)
```

```
0.9710144927536232
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.2, random_state=42)

scaler = StandardScaler()

scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
# Las dos anteriores instrucciones serían equivalentes a:
# X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)
```

```
bagging_clf = BaggingClassifier(
    DecisionTreeClassifier(), n_estimators=200,
    max_samples=100, bootstrap=True, n_jobs=-1)

bagging_clf.fit(X_train, y_train)

y_pred10 = bagging_clf.predict(X_test[:10])
y_pred10
```

```
array(['Chinstrap', 'Chinstrap', 'Gentoo', 'Chinstrap', 'Gentoo',
```

```
       'Gentoo', 'Gentoo', 'Gentoo', 'Adelie', 'Gentoo'], dtype=object)
```

```
y_test[:10]
```

```
194    Chinstrap
157    Chinstrap
225       Gentoo
208    Chinstrap
318       Gentoo
329       Gentoo
319       Gentoo
260       Gentoo
114       Adelie
220       Gentoo
Name: species, dtype: object
```

```
y_pred = bagging_clf.predict(X_test)
```

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(), max_samples=100,
                  n_estimators=200, n_jobs=-1)
```

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(), max_samples=100,
                  n_estimators=200, n_jobs=-1)
```

```
print(classification_report(y_pred, y_test)) # Una visión más detallada del rendimiento por c
```

```
              precision    recall  f1-score   support

      Adelie       1.00      0.97      0.98        33
   Chinstrap       0.94      1.00      0.97        15
      Gentoo       1.00      1.00      1.00        21

    accuracy                           0.99        69
   macro avg       0.98      0.99      0.98        69
weighted avg       0.99      0.99      0.99        69
```

```
confusion_matrix(y_pred, y_test)
```

```
array([[32,  1,  0],
       [ 0, 15,  0],
       [ 0,  0, 21]])
```

```
accuracy_score(y_pred, y_test)
```

```
0.9855072463768116
```

```
df_num = penguins.drop('species', axis=1) # Eliminar variable categórica
```

```
df_num.head()

X = df_num.drop('body_mass_g',axis=1)
y = df_num['body_mass_g']

# Particionado en entrenamiento + test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
ada_reg = AdaBoostRegressor(
    DecisionTreeRegressor(max_depth=2), n_estimators=100,
    learning_rate=0.5)

ada_reg.fit(X_train, y_train)
```

```
AdaBoostRegressor(base_estimator=DecisionTreeRegressor(max_depth=2),
                  learning_rate=0.5, n_estimators=100)
```

```
y_pred = ada_reg.predict(X_test)

mse = mean_squared_error(y_pred, y_test)
rmse = np.sqrt(mse)
rmse
```

```
407.0114669170131
```

```
penguins.body_mass_g.mean()/397.39171939550465
```

```
10.573331503626804
```

```
df_num = penguins.drop('species', axis=1) # Eliminar variable categórica
df_num.head()

X = df_num.drop('body_mass_g',axis=1)
y = df_num['body_mass_g']

# Particionado en entrenamiento + test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4, random_state = 42)
ada_reg = AdaBoostRegressor(
    DecisionTreeRegressor(max_depth=2), n_estimators=100,
    learning_rate=0.5)

ada_reg.fit(X_train, y_train)
```

```
AdaBoostRegressor(base_estimator=DecisionTreeRegressor(max_depth=2),
                  learning_rate=0.5, n_estimators=100)
```

```
y_pred = ada_reg.predict(X_test)

mse = mean_squared_error(y_pred, y_test)
```

```
rmse = np.sqrt(mse)
rmse
```

```
373.7301469260958
```

```
penguins.body_mass_g.mean()/377.55171876274784
```

```
11.128950491165105
```

✓ 0 s completado a las 16:39