In [1]:
```
#Load Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import tree
```

In [2]:
```
#Load Data
df = pd.read_csv('./ricedataset.csv')
df.head()
```

Out[2]:

	AREA	PERIMETER	MAJORAXIS	MINORAXIS	ECCENTRICITY	CONVEX_AREA	EXTENT	CLASS
0	15231	525.578979	229.749878	85.093788	0.928882	15617	0.572896	Cammeo
1	14656	494.311005	206.020065	91.730972	0.895405	15072	0.615436	Cammeo
2	14634	501.122009	214.106781	87.768288	0.912118	14954	0.693259	Cammeo
3	13176	458.342987	193.337387	87.448395	0.891861	13368	0.640669	Cammeo
4	14688	507.166992	211.743378	89.312454	0.906691	15262	0.646024	Cammeo

In [3]:
```
#Identify number of Classes (i.e. Species)
df.CLASS.unique()
```

Out[3]:
```
array(['Cammeo', 'Osmancik'], dtype=object)
```

In [4]:
```
#Key Statistics
df.describe()
```

Out[4]:

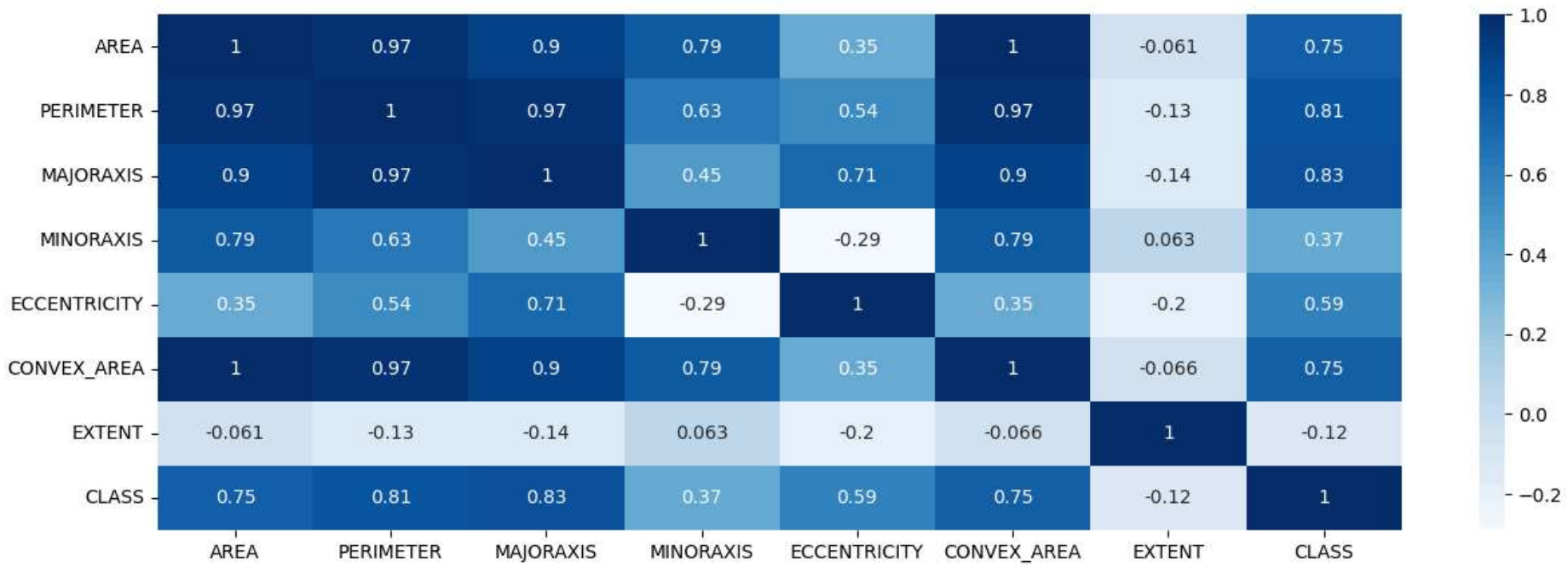
	AREA	PERIMETER	MAJORAXIS	MINORAXIS	ECCENTRICITY	CONVEX_AREA	EXTENT
count	3810.000000	3810.000000	3810.000000	3810.000000	3810.000000	3810.000000	3810.000000
mean	12667.727559	454.239180	188.776222	86.313750	0.886871	12952.496850	0.661934
std	1732.367706	35.597081	17.448679	5.729817	0.020818	1776.972042	0.077239
min	7551.000000	359.100006	145.264465	59.532406	0.777233	7723.000000	0.497413
25%	11370.500000	426.144752	174.353855	82.731695	0.872402	11626.250000	0.598862
50%	12421.500000	448.852493	185.810059	86.434647	0.889050	12706.500000	0.645361
75%	13950.000000	483.683746	203.550438	90.143677	0.902588	14284.000000	0.726562
max	18913.000000	548.445984	239.010498	107.542450	0.948007	19099.000000	0.861050

In [5]:
```
#check for null values
df.isnull().sum()
```

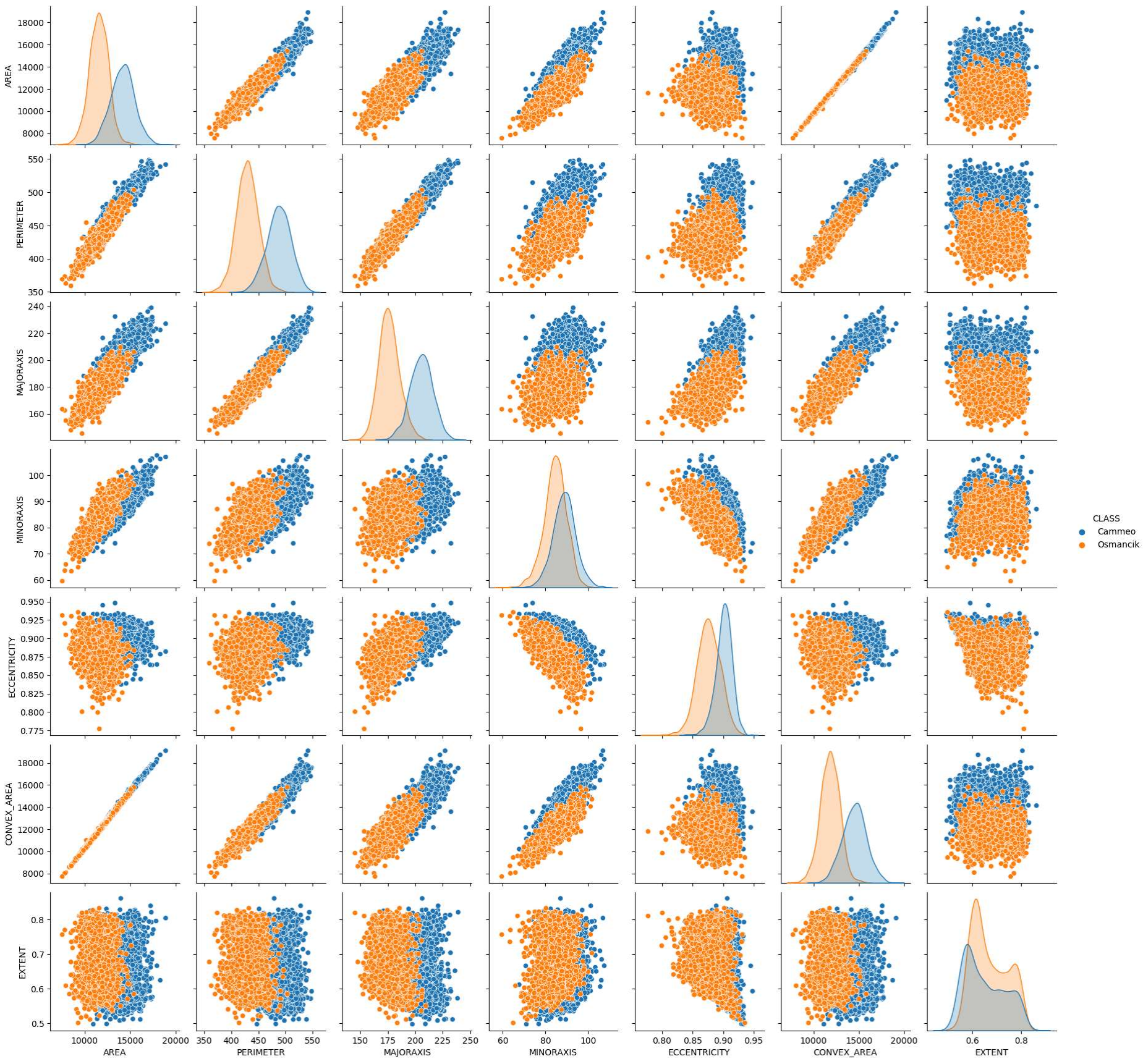
Out[5]:
```
AREA            0
PERIMETER       0
MAJORAXIS       0
MINORAXIS       0
ECCENTRICITY    0
CONVEX_AREA     0
EXTENT          0
CLASS           0
dtype: int64
```

In [6]:
```
#Visualization of Correlations
rice = df.copy()
rice
rice['CLASS'] = rice['CLASS'].map({'Cammeo':1,'Osmancik':0})
fig = plt.figure(figsize=(15,5))
sns.heatmap(rice.corr(),annot=True,cmap="Blues")
```

Out[6]:
```
<AxesSubplot: >
```



In [7]:
```
sns.pairplot(df, hue='CLASS');
```



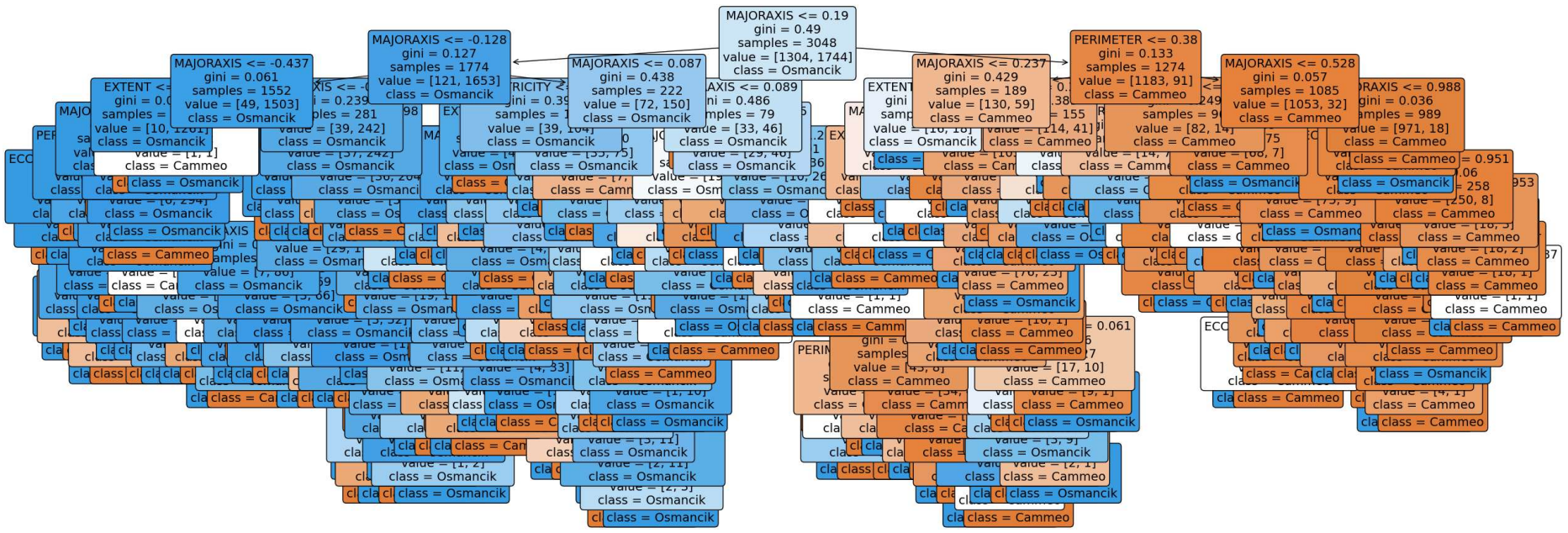
In [8]:
```
#Create x and y variables
X = df.drop('CLASS',axis=1).to_numpy()
y = df['CLASS'].to_numpy()
x = df.drop('CLASS',axis=1)
Y = df['CLASS']

feature_names = x.columns
labels = Y.unique()

#Create Train and Test datasets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,test_size = 0.20,random_state=100)

#Scale the data
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
x_train2 = sc.fit_transform(X_train)
x_test2 = sc.transform(X_test)
```

In [9]:
```
#Script for Decision Tree
for name,method in [('DT', DecisionTreeClassifier(random_state=100))]:
    method.fit(x_train2,y_train)
    predict = method.predict(x_test2)
    target_names=['Cammeo','Osmancik']
    print('\nEstimator: {}'.format(name))
    print(confusion_matrix(y_test,predict))
    print(classification_report(y_test,predict,target_names=target_names))
```

```
Estimator: DT
[[291  35]
 [ 45 391]]
              precision    recall  f1-score   support

      Cammeo       0.87      0.89      0.88       326
    Osmancik       0.92      0.90      0.91       436

    accuracy                           0.90       762
   macro avg       0.89      0.89      0.89       762
weighted avg       0.90      0.90      0.90       762
```

In [10]:
```
#plt the figure, setting a black background for default values and random state 100
plt.figure(figsize=(30,10))
#create the tree plot
a = tree.plot_tree(method,
                   #use the feature names stored
                   feature_names = feature_names,
                   #use the class names stored
                   class_names = labels,
                   rounded = True,
                   filled = True,
                   fontsize=14)
#show the plot
plt.show()
```

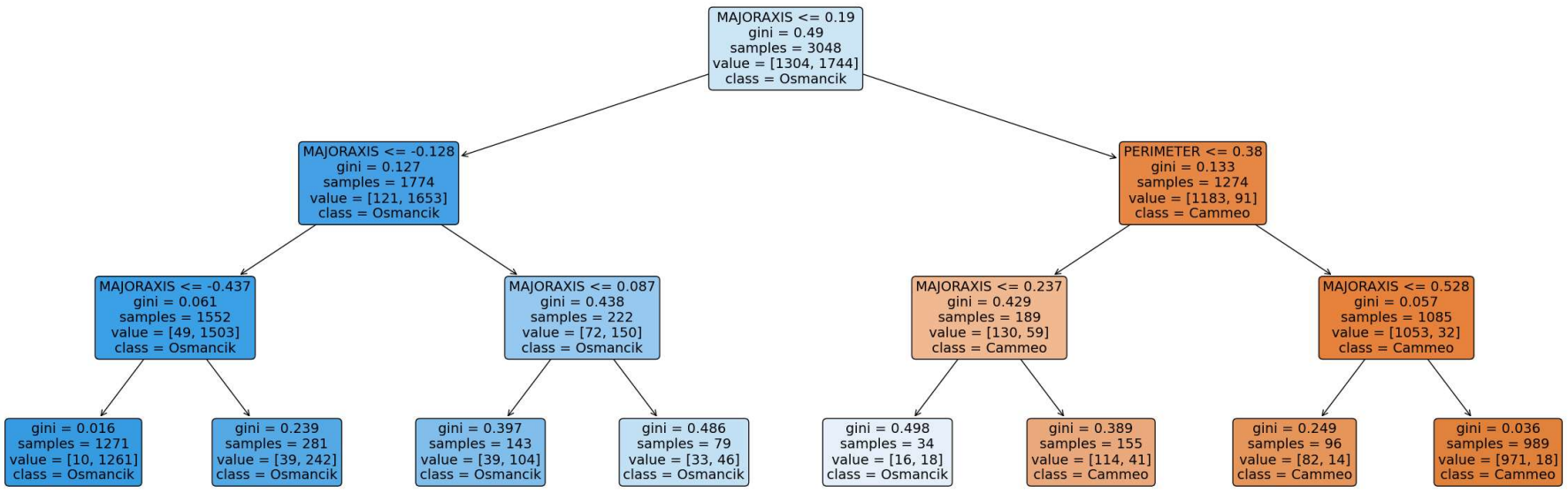


In [11]:
```
#Script for Decision Tree with max depth of 3
for name,method in [('DT', DecisionTreeClassifier(max_depth= 3, random_state=100))]:
    method.fit(x_train2,y_train)
    predict = method.predict(x_test2)
    target_names=['Cammeo','Osmancik']
    print('\nEstimator: {}'.format(name))
    print(confusion_matrix(y_test,predict))
    print(classification_report(y_test,predict,target_names=target_names))
```

```
Estimator: DT
[[291  35]
 [ 24 412]]
              precision    recall  f1-score   support

      Cammeo       0.92      0.89      0.91       326
    Osmancik       0.92      0.94      0.93       436

    accuracy                           0.92       762
   macro avg       0.92      0.92      0.92       762
weighted avg       0.92      0.92      0.92       762
```

In [12]:
```
#plt the figure, setting a black background for max depth 3
plt.figure(figsize=(30,10))
#create the tree plot
a = tree.plot_tree(method,
                   #use the feature names stored
                   feature_names = feature_names,
                   #use the class names stored
                   class_names = labels,
                   rounded = True,
                   filled = True,
                   fontsize=14)
```

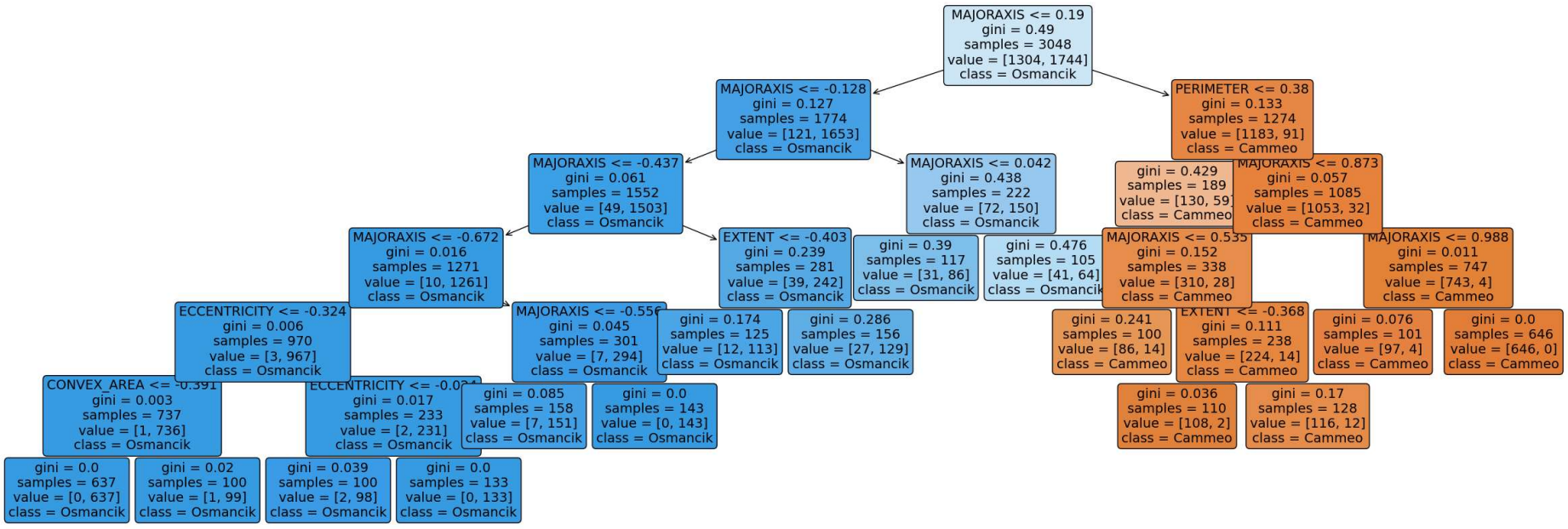
```
#show the plot
plt.show()
```



In [13]:
```
#Script for Decision Tree min samples leaf

for name,method in [('DT', DecisionTreeClassifier(min_samples_leaf=100,random_state=100))]:
    method.fit(x_train2,y_train)
    predict = method.predict(x_test2)
    target_names=['Cammeo','Osmancik']
    print('\nEstimator: {}'.format(name))
    print(confusion_matrix(y_test,predict))
    print(classification_report(y_test,predict,target_names=target_names))
```

```
Estimator: DT
[[293  33]
 [ 26 410]]
              precision    recall  f1-score   support

      Cammeo       0.92      0.90      0.91       326
    Osmancik       0.93      0.94      0.93       436

    accuracy                           0.92       762
   macro avg       0.92      0.92      0.92       762
weighted avg       0.92      0.92      0.92       762
```

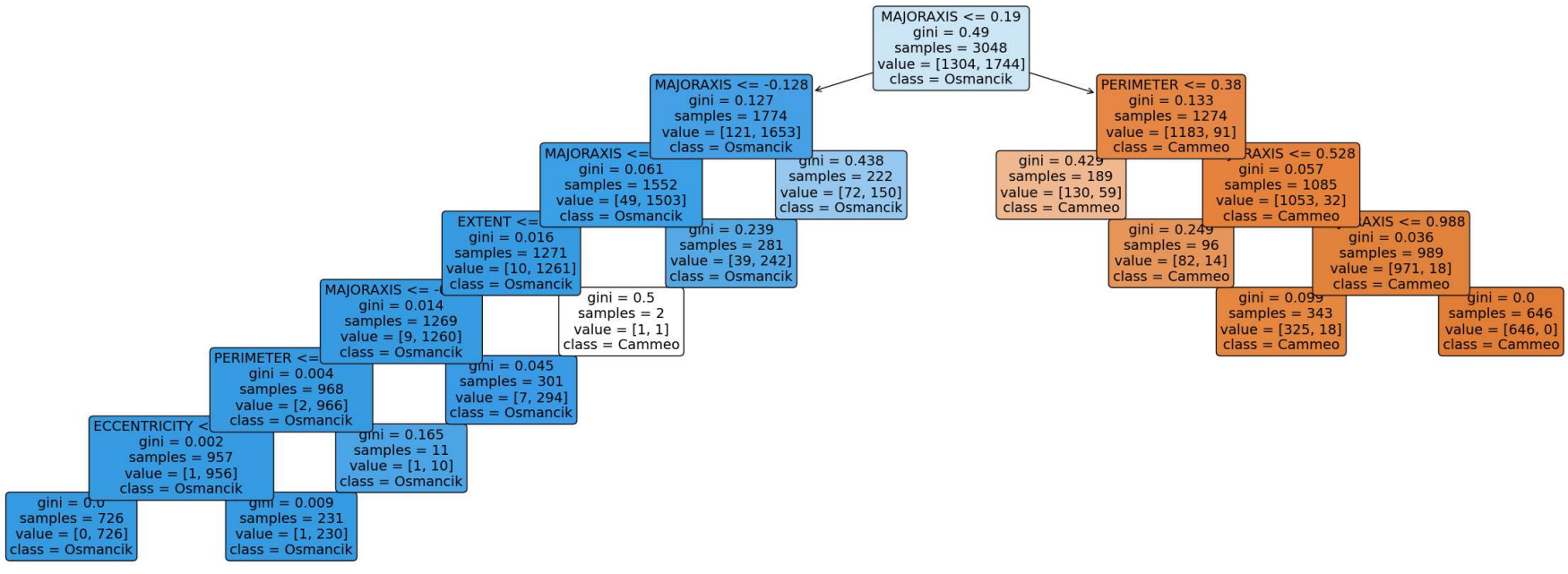
In [14]:
```
#plt the figure, setting a black background for min samples leaf
plt.figure(figsize=(30,10))
#create the tree plot
a = tree.plot_tree(method,
                   #use the feature names stored
                   feature_names = feature_names,
                   #use the class names stored
                   class_names = labels,
                   rounded = True,
                   filled = True,
                   fontsize=14)
#show the plot
plt.show()
```



In [15]:
```
#Script for Decision Tree with min saples split

for name,method in [('DT', DecisionTreeClassifier(min_samples_split=400,random_state=100))]:
    method.fit(x_train2,y_train)
    predict = method.predict(x_test2)
    target_names=['Cammeo','Osmancik']
    print('\nEstimator: {}'.format(name))
    print(confusion_matrix(y_test,predict))
    print(classification_report(y_test,predict,target_names=target_names))
```

```
Estimator: DT
[[293  33]
 [ 26 410]]
              precision    recall  f1-score   support

      Cammeo       0.92      0.90      0.91       326
    Osmancik       0.93      0.94      0.93       436

    accuracy                           0.92       762
   macro avg       0.92      0.92      0.92       762
weighted avg       0.92      0.92      0.92       762
```

In [16]:
```
#plt the figure, setting a black background for min samples split
plt.figure(figsize=(30,10))
#create the tree plot
a = tree.plot_tree(method,
                   #use the feature names stored
                   feature_names = feature_names,
                   #use the class names stored
                   class_names = labels,
                   rounded = True,
                   filled = True,
                   fontsize=14)
#show the plot
plt.show()
```



In [ ]:
```
```