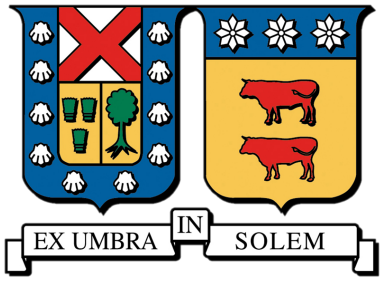


Universidad Técnica Federico Santa María

MAT-281

Tarea 2

Aplicaciones de la Matemática a la Ingeniería

Autor:
Rodrigo Serrano Pérez

2023-2

1. Presentación del Problema

† **Contexto:** Considere el conjunto de datos del archivo rice.txt. Este es un problema de clasificación binaria donde granos de arroz pueden ser clasificados en alguna de las siguientes especies: Cammeo o Osmancik. Se tiene un total de 3810 imágenes de granos de arroz. Estas imágenes fueron procesadas y se obtuvieron 7 características morfológicas (covariables) para cada grano de arroz:

1. Área: Devuelve el número de píxeles dentro de los límites del grano de arroz.
2. Perímetro: Calcula la circunferencia a partir de la distancia entre píxeles alrededor de los límites del grano de arroz.
3. Longitud del Eje Mayor: La línea más larga que se puede trazar en el grano de arroz.
4. Longitud del Eje Menor: La línea más corta que se puede trazar en el grano de arroz.
5. Excentricidad: Mide qué tan redonda es la elipse que tiene los mismos momentos que el grano de arroz.
6. Área Convexa: Devuelve la cantidad de píxeles de la cáscara convexa más pequeña de la región formada por el grano de arroz.
7. Extensión: Devuelve la proporción de la región formada por el grano de arroz en relación con los píxeles del cuadro delimitador.

† ¿Qué se espera de esta tarea? El propósito de esta tarea consiste en comparar el rendimiento de los métodos de vecinos cercanos y Support Vector Machine (SVM). Para lograr este objetivo, usted debe realizar estudios de validación cruzada. Tenga en cuenta la variación del número de vecinos en el caso de los vecinos cercanos y la elección del kernel en el caso de SVM.

Para aquello, proceda de la siguiente forma:

1. Haga un análisis descriptivo de los datos (reporte indicadores y gráficos para tener una perspectiva general del problema).
2. Luego, proporcione una explicación detallada de los procedimientos empleados en cada uno de estos métodos y presente las conclusiones derivadas de dichos análisis.

Es importante que sus estudios estén acompañados de matrices de confusión, gráficos apropiados, indicadores de precisión, etc. Debe entregar un informe en un formato similar a la tarea 1 .

2. Análisis Descriptivo de los datos

2.1. Cantidades y Estadísticos

Desde el archivo rice.txt, al desglosar los datos para luego realizar una clasificación de los tipos de arroces. Se obtuvo que se dispone de Conteo de Clases:

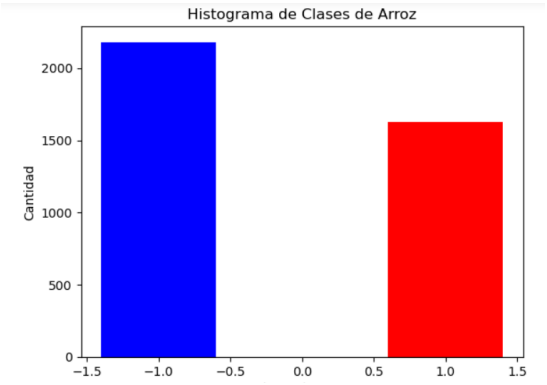


Figura 1: Azul: Osmancik, Rojo: Cammeo

De donde se tienen además, se realizaron los cálculos de estadísticos para la muestra de los datos, de lo anterior se obtuvo que:

```
Resumen Estadístico de las Características:
               Area    Perimetro  Longitud eje mayor  Longitud eje menor  \
count   3810.000000  3810.000000         3810.000000         3810.000000
mean   12667.727559   454.239180          188.776222           86.313750
std     1732.367706    35.597081           17.448679            5.729817
min     7551.000000   359.100006          145.264465           59.532406
25%    11370.500000   426.144753          174.353855           82.731695
50%    12421.500000   448.852493          185.810059           86.434647
75%    13950.000000   483.683746          203.550438           90.143677
max    18913.000000   548.445984          239.010498          107.542450

       Excentricidad  Area convexa    Extension
count    3810.000000   3810.000000  3810.000000
mean        0.886871  12952.496850     0.661934
std         0.020818   1776.972042     0.077239
min         0.777233   7723.000000     0.497413
25%         0.872402  11626.250000     0.598862
50%         0.889050  12706.500000     0.645361
75%         0.902588  14284.000000     0.726562
max         0.948007  19099.000000     0.861050

Conteo de Clases:
Osmancik    2180
Cammeo      1630
```

Figura 2: Estadísticos y Conteo

2.2. Histogramas Respectivos

Luego, se realizaron histogramas respecto la frecuencia y los valores de cada una de las características de los arroces, de lo anterior, obtuvimos que:

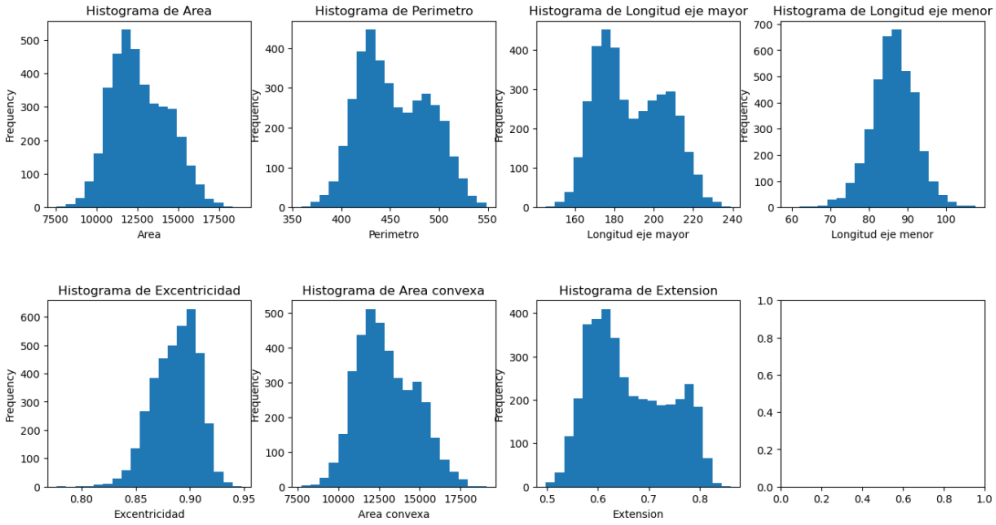


Figura 3: Histogramas

Sin embargo, lo anterior sólo plasma la frecuencia de las cantidades para cada una de las características. Lo cual a priori, no nos entrega detalles sobre la diferencia de estos valores respecto las clases de arroces. Para lo anterior, se realizó además otro histograma con el mismo esquema pero dividiendo por clase al arroz. De donde se obtuvo el histograma que se presenta en la siguiente página.

De este, se puede notar que en el caso de la extensión del arroz no existe grandes diferencias respecto la clase a la cual pertenece. A diferencia por ejemplo de la longitud del eje mayor. Pero cabe destacar que en el caso de los estadísticos que se calcularon en el inciso anterior, existe una gran diferencia respecto estos para las clases de arroz. Por ejemplo, se nota que en el caso del perímetro y área, Cammeo presenta un mayor valor en su promedio y mediana respecto a Osmancik, lo cual es consistente con la mátematica pues a mayor área mayor será el perímetro (en el caso donde estamos trabajando más menos con la misma figura geométrica).

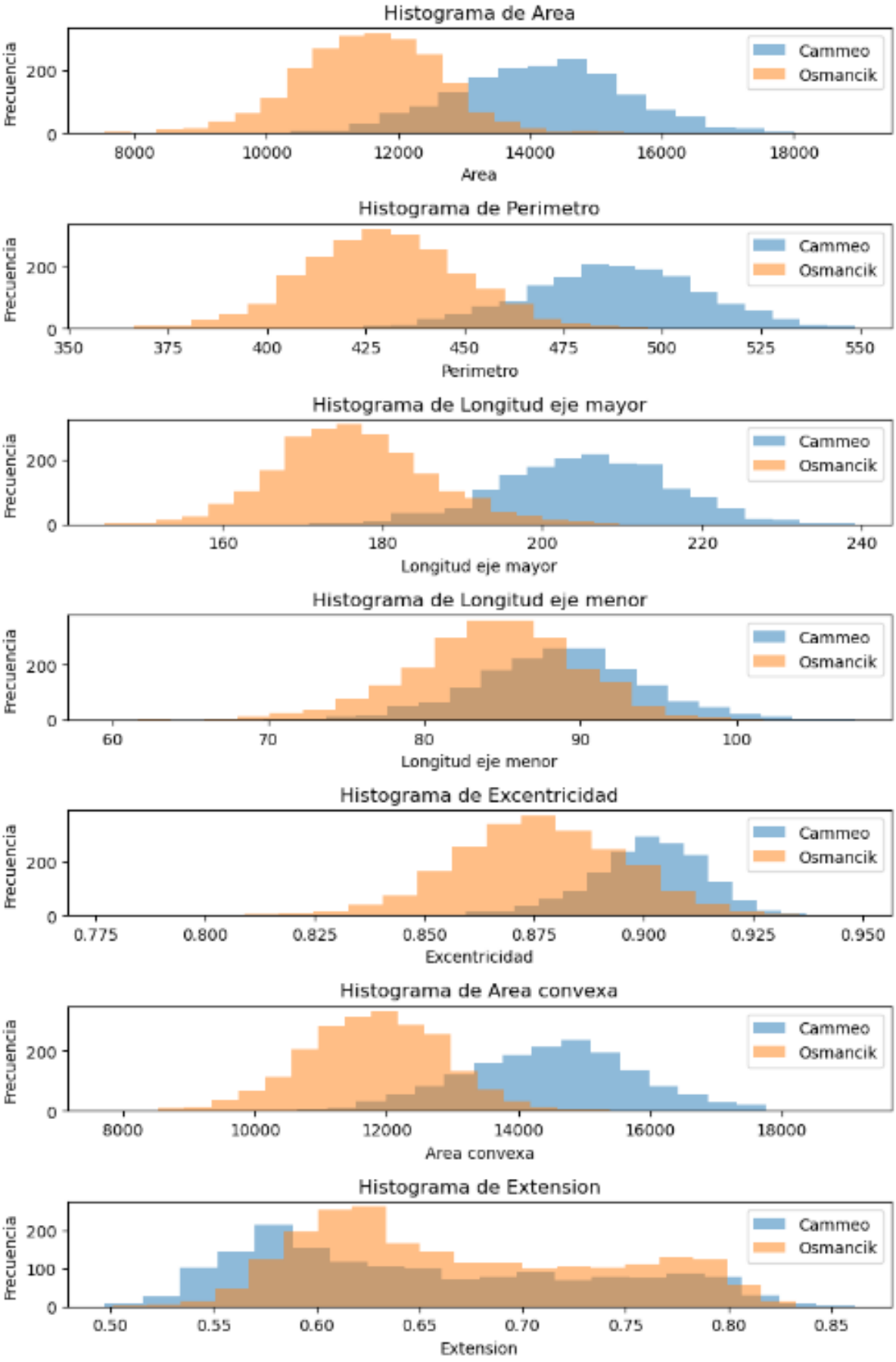


Figura 4: Histogramas por Clase

2.3. Dispersión de los Datos

Adicionalmente, llevamos a cabo una representación gráfica de la dispersión entre las covariables y sus distribuciones para evaluar la magnitud de las diferencias entre las distintas categorías de arroz. Los resultados de este análisis se presentan a continuación:

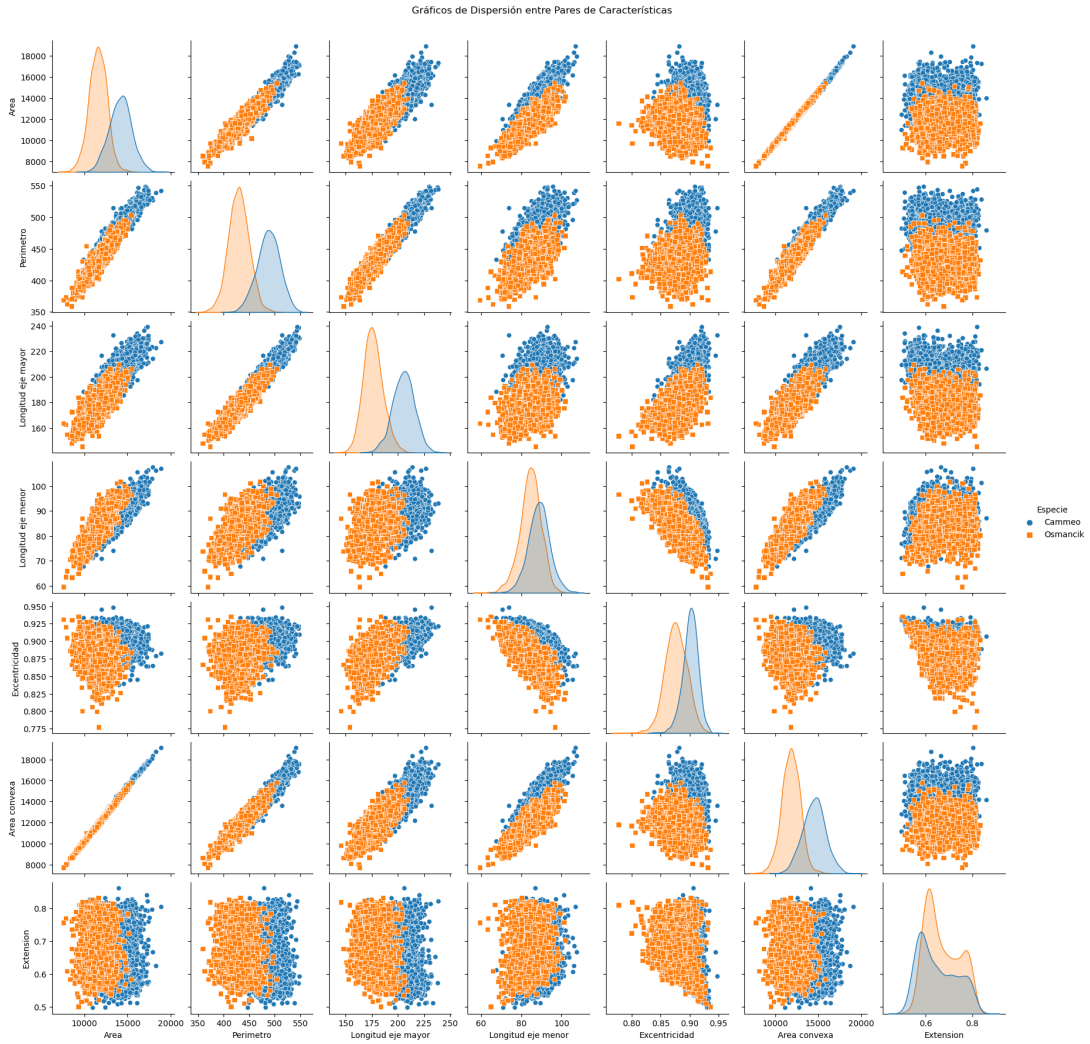


Figura 5: Dispersión de los Datos

Observe los gráficos y notará que en la mayoría de ellos, existe una notable superposición de puntos entre las dos categorías de arroz en relación a sus diversas características. Esta observación inicial sugiere que, por ejemplo, cuando se trata de medir la excentricidad o su extensión, no se aprecian diferencias significativas que permitan clasificar el arroz de manera clara en una u otra categoría como Cammeo u Osmancik. Por lo cual, las diferencias en la excentricidad no parecen ser un factor distintivo para determinar a qué clase pertenece el arroz.

2.4. Matrices de Correlación

Es también importante calcular la matriz de correlación para plasmar entre los datos. Al realizar este ejecutamiento en python, se obtuvo lo que se grafica a continuación:

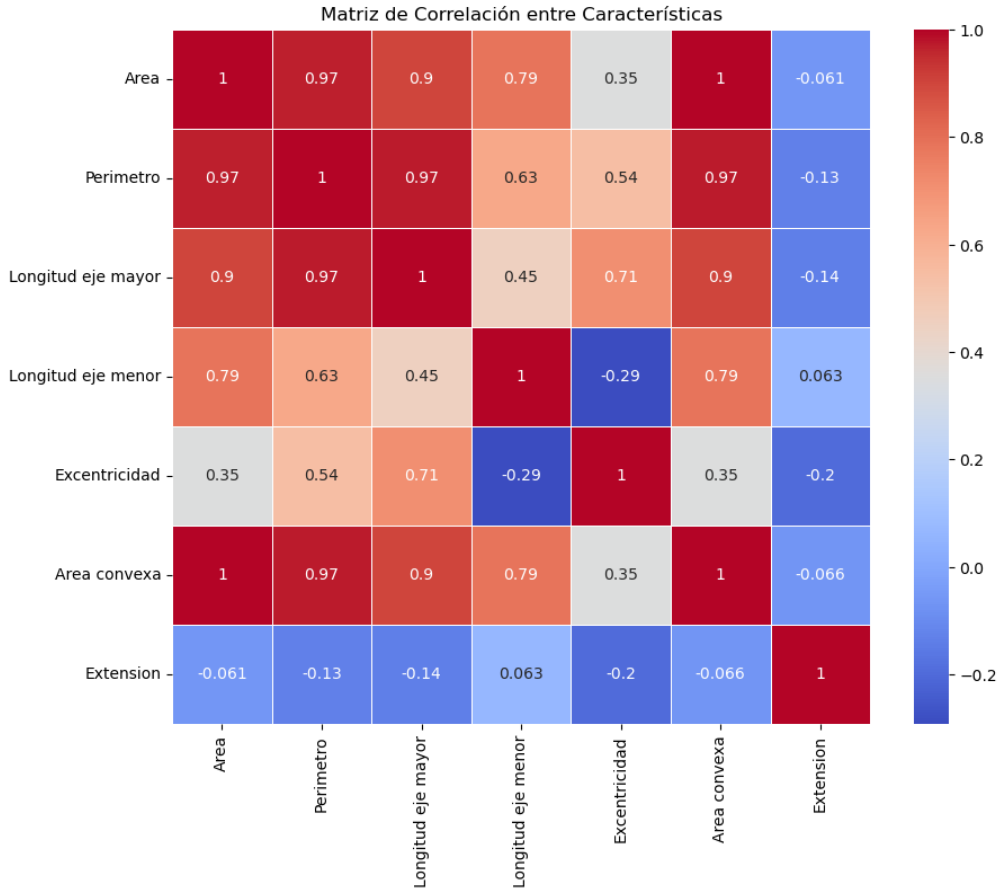


Figura 6: Matriz de Correlaciones

De lo anterior, podemos inferir que la extensión del arroz o su excentricidad en si, no interfiere o no se correlaciona respecto las demás características presentes en una muestra de arroz a como lo puede ser el área o su perímetro. Lo que concuerda con lo mostrado en el gráfico de dispersión presentadoa anteriormente. Sin embargo, veamos como se comporta la correlación de las características en comparación a cada clase de arroz. Para esto, podemos dividir el dataframe en dos dataframes usando:

```
path = "C:/Users/asus/Desktop/cosas_R/rice.txt"
df = pd.read_csv(path, sep=",", header=None, names=["Area", "Perimetro",
    "Longitud eje mayor", "Longitud eje menor", "Excentricidad", "Area
    convexa", "Extension", "Especie"])

df_cammeo = df[df["Especie"] == "Cammeo"]
df_osmancik = df[df["Especie"] == "Osmancik"]
```

Para así luego poder calcular la misma matriz pero para cada una de las clases de arroz. Entonces, se obtuvo lo que se muestra en la siguiente página.

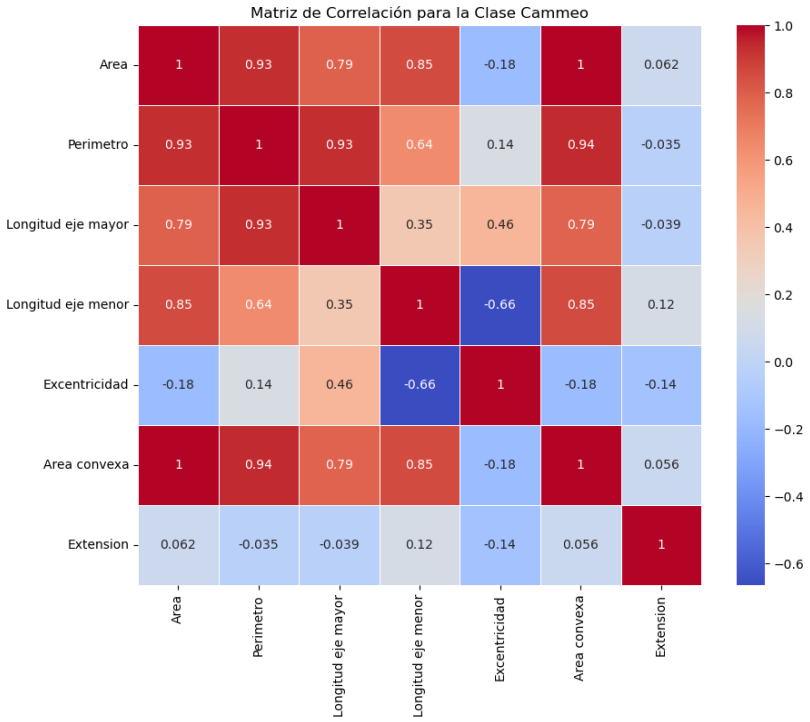


Figura 7: Matriz de Correlación clase: Cammeo

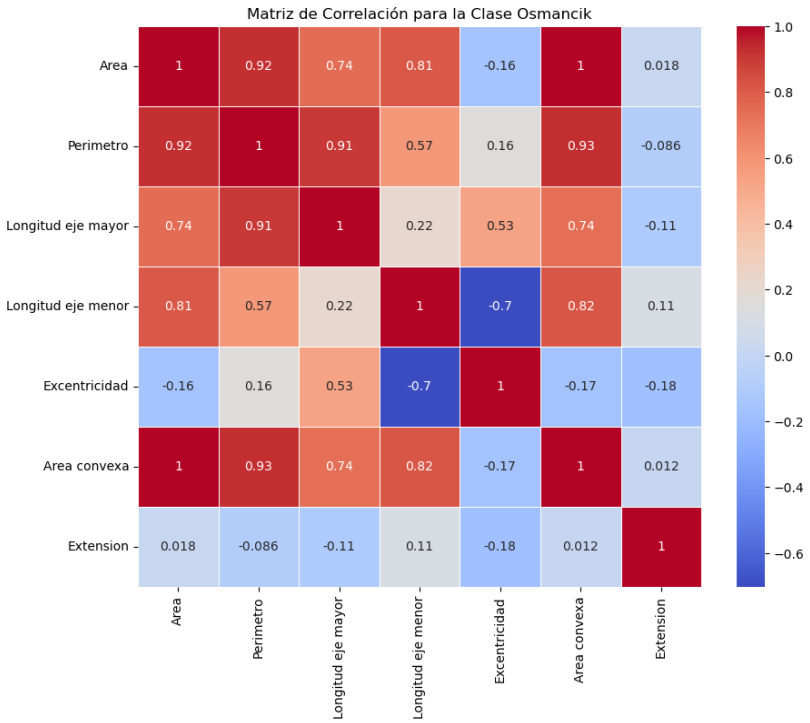


Figura 8: Matriz de Correlación clase: Osmancik

Notamos que la diferencia entre las correlaciones para los tipos de arroces no exceden en su mayoría el factor $0,1$. Lo cual podría sugerir que no es tan importante la clase de arroz para la correlación entre las características.

3. Preparación de los datos

Para esta sección, se realizará una preparación de los datos diviendo una parte la cual llamaremos el conjunto de entrenamiento, el cual tendrá un factor de $0,25$. Esto, con el fin de cuando más adelante realicemos los métodos de KNN o SVM, podamos ver cual modelo se ajusta de mejor manera respecto al otro. Para esto, considere el siguiente extracto de código:

```
    import pandas as pd
    from sklearn.model_selection import train_test_split
    data["Especie"] = data["Especie"].replace({"Osmancik": -1, 'Cammeo': 1})

    # Dividir los datos en caracteristicas (X) y etiquetas (y)
    X = data.drop("Especie", axis=1)
    y = data["Especie"]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
        random_state=45)
```

Donde los nombres de las train o test definen los datos de entrenamiento y de testeo para este proceso.

Luego de lo anterior, queremos estandarizar los datos para obtener mejores resultados en nuestros testeos. Por lo cual, utilizando *.StandardScaler*, procedemos a realizar lo anterior para los datos de testeo y de entrenamiento. Mediante:

```
    X_scaler = preprocessing.StandardScaler()
    cols_to_normalize = X_train.columns.tolist()
    X_train[cols_to_normalize]=
        X_scaler.fit_transform(X_train[cols_to_normalize])
    X_test[cols_to_normalize]= X_scaler.fit_transform(X_test[cols_to_normalize])
```

Ahora tenemos nuestros datos listos para emplear los métodos de SVM y KNN para este problema. Lo cual viene en la próxima sección.

4. Método KNN

4.1. Implementación

Procediendo a la creación del modelo KNN, se obtiene lo siguiente:

```
    model_base_knn = KNeighborsClassifier(algorithm='ball_tree')
    param_grid_knn = {
        'n_neighbors' : [1, 3, 5, 8, 10, 12, 15, 30, 100]
    }
    best_knn = GridSearchCV(model_base_knn,
                            param_grid_knn,
                            cv = 5, # cantidad de divisiones al train set
                            scoring = 'f1' # 'accuracy', 'precision', 'recall',
                                'f1'
                            )
    best_knn.fit(X_train, y_train)
```

En donde "GridSearchCV", su función principal es encontrar el número óptimo de vecinos ("neighbors") para el modelo de "ball tree". Esto se logra dividiendo los datos en 5 conjuntos, lo que es controlado por el parámetro "cv". La evaluación del rendimiento se realiza utilizando el puntaje F1, que proporciona una forma de medir para la evaluación. El resultado final será un gráfico que representa el mejor parámetro encontrado a través de este proceso.

Además, es importante destacar que "GridSearchCV", es una técnica de búsqueda que evalúa diferentes valores de hiperparámetros para encontrar la configuración óptima del modelo. Esto ayuda a optimizar el rendimiento del modelo de "ball tree.[a]l encontrar la cantidad de vecinos que maximiza el F1 score, lo que a su vez puede mejorar la precisión y eficacia del modelo en la clasificación de datos. Mientras que "ball tree.[es] una estructura de datos que acelera la búsqueda de vecinos cercanos en espacios de alta dimensión, útil en algoritmos como k-NN.

F1 score, es representado mediante una fórmula que utiliza la sensibilidad y la precisión del modelo. En efecto, lo anterior, se plasma por, si se definen los valores:

1. TP= El número total de los **Positivos Acertados**
2. FP= El número total de los falsos **Positivos**
3. FN= El número total de los falsos **Negativos**

$$F1 \text{ Score} = \frac{2 \cdot \text{Precisión} \cdot \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}} = \frac{2 \cdot \frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}}$$

De lo anterior, buscaremos que nuestro valor "F1 Score", tenga una magnitud cercana a "1". Pues nos indicaría que el modelo es balanceado. De otra manera, si este tiende a 0, nos indicaría que el método no se ajusta o no está funcionando de forma correcta.

Con el preámbulo anterior, al cálcular el número de vecinos cercanos óptimo y el valor F1 Score, mediante:

```
    best_knn.best_params_
    f1_score_knn = f1_score(y_test, y_pred_knn)
    f1_score_knn
```

Fue de **15** en el caso de los vecinos y de **0,911** en el caso de F1 Score.

4.2. Resultados Obtenidos

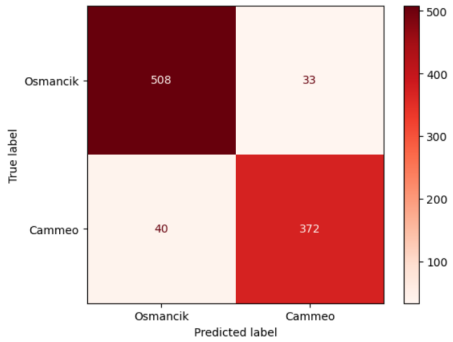
Al realizar un reporte de clasificación, se obtuvo que:

```
print(classification_report(y_test, y_pred_knn, target_names=["Cammeo", "Osmancik"]))
```

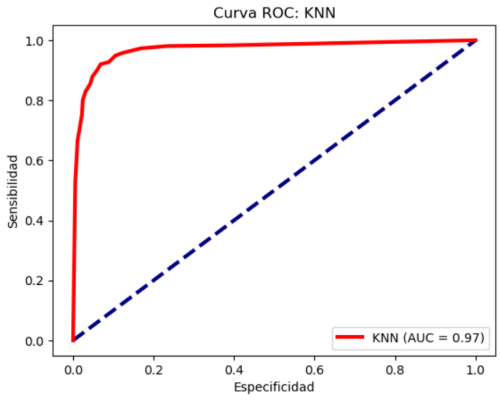
	precision	recall	f1-score	support
Cammeo	0.93	0.94	0.93	541
Osmancik	0.92	0.90	0.91	412
accuracy			0.92	953
macro avg	0.92	0.92	0.92	953
weighted avg	0.92	0.92	0.92	953

En el principio de este documento, se habló de que existían características que no tenían gran impacto en la información. Como lo fue el caso de la **Extensión**. Sin embargo, al recalcular los parámetros de F1-Score, no se obtuvo grandes mejorías respecto a lo anterior. Por lo cual, se determinó por desechar esa idea y quedarnos con lo anterior.

Utilizando este método, la Matriz de Confusión viene dada por:



Donde vemos que en comparación a la cantidad del Predicto v/s Real, el método se comporta bastante bien y posee un error menos a $0,1$, lo cual concuerda con nuestro paramétro F1-Score calculado anteriormente. De igual forma, se calculó una curva ROC para poder consolidar lo anterior para luego realizar una mejor comparación entre ambos modelos. El resultado es el siguiente:



La curva ROC se asemeja más a la diagonal superior de un cuadrado en lugar de acercarse a la línea segmentada que se comporta como una función identidad. Esto indica que el modelo tiene un rendimiento muy sólido, ya que el AUC de 0.97 sugiere una excelente capacidad para discriminar entre las clases, lo cual es preferible en la evaluación de modelos de clasificación.

5. Método de SVM

5.1. Implementación

El código utilizado fue:

```
    model_base_svm = SVC()
    param_grid_svm = {
        'C' : [0.1, 1, 10],
        'kernel' : ['linear', 'rbf', 'poly'],
        'degree' : [2, 3],
    }
    best_svm = GridSearchCV(model_base_svm,
                            param_grid_svm,
                            cv = 5,
                            scoring = 'f1'
                            )
    best_svm.fit(X_train, y_train)
```

De lo anterior, utilizando el mismo conjunto de entrenamiento y también GridSearchCV con F1-Score que presentamos en la sección anterior. En donde además Ç", en SVM controla la flexibilidad del modelo. Valores más altos hacen que el modelo sea más ajustado a los datos, mientras que valores bajos permiten errores de clasificación. Por otro lado, 'degree' se aplica a kernels polinómicos en SVM y controla la complejidad de la transformación polinómica. Valores más altos hacen que el modelo sea más complejo y propenso al sobreajuste.

5.2. Resultados

Los valores óptimos que se obtuvieron con este método se detallan:

```
: best_svm.best_params_
: {'C': 10, 'degree': 2, 'kernel': 'linear'}
```

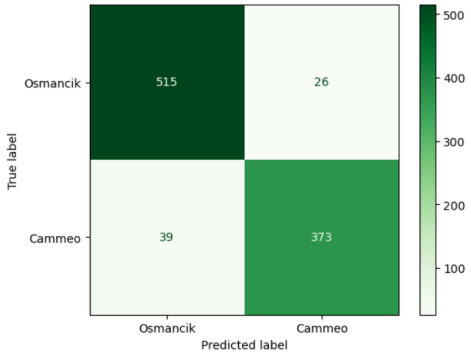
Donde el "Kernel"que debería tener un carácter polinomico, fue lineal lo cual concuerda. Ahora, el valor de F1-Score, fue de **0,920**.

Al realizar el reporte de clasificación para el método de SVM, lo que se obtuvo fue:

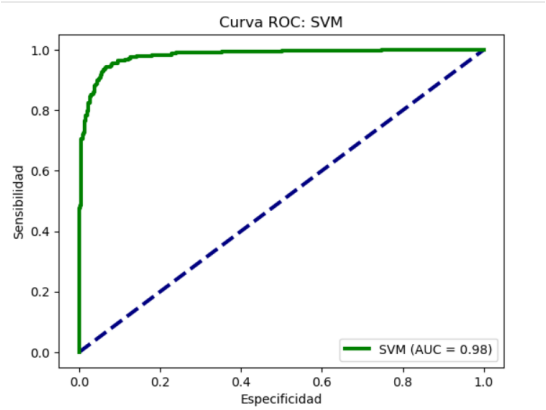
```
print(classification_report(y_test, y_pred_svm, target_names=["Cammeo", "Osmancik"]))
```

	precision	recall	f1-score	support
Cammeo	0.93	0.95	0.94	541
Osmancik	0.93	0.91	0.92	412
accuracy			0.93	953
macro avg	0.93	0.93	0.93	953
weighted avg	0.93	0.93	0.93	953

Al igual que en la sección anterior, ocurrió algo similar respecto a la eliminación de dimensiones para el problema. De forma que se desechó también esa idea. Calculando la matriz de Confusión, se obtuvo que:



De donde también podemos inferir que el modelo se ajusta bastante y tiene una buena predibilidad respecto los datos. Lo cual también es consistente con nuestro parámetro F1-Score. Ahora, al hacer un "plot"de la curva ROC, se tiene que:



La cual también tiene un buen comportamiento pues en burdas palabras toma la forma de la diagonal superior de un cuadrado y no se asemeja o acerca a una función indentidad como es la linea segmentada.

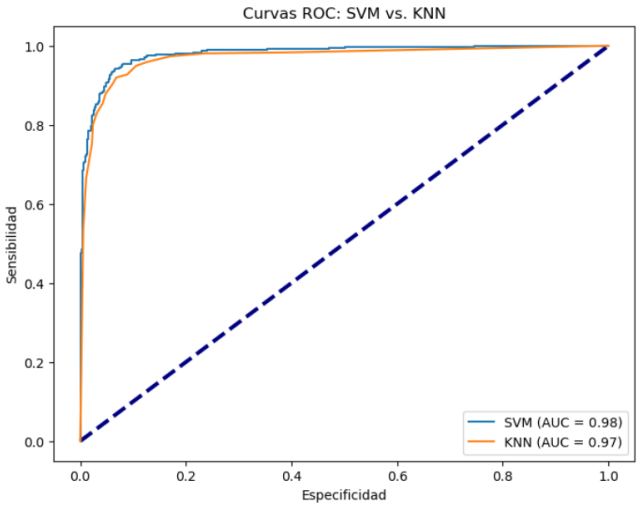
6. Comparaciones y Conclusiones

A continuación, procederemos a realizar una comparación de los valores obtenidos mediante los métodos SVM y KNN con el propósito de determinar cuál de estos enfoques es más adecuado para abordar el problema en cuestión.

Esta comparación nos permitirá evaluar y contrastar el rendimiento de ambos modelos en términos de sus métricas de evaluación, como la precisión, el F1-Score, o cualquier otro indicador relevante como los que veremos más adelante. Una vez analizados estos resultados, podremos tomar una decisión informada sobre cuál de los dos métodos es más efectivo para resolver el problema.

6.1. Comparación curvas ROC

En primera instancia, evaluemos la superposición de las curvas ROC, de donde sabemos, que mientras más se acercan a la identidad, tiene un peor "desempeño". En efecto:

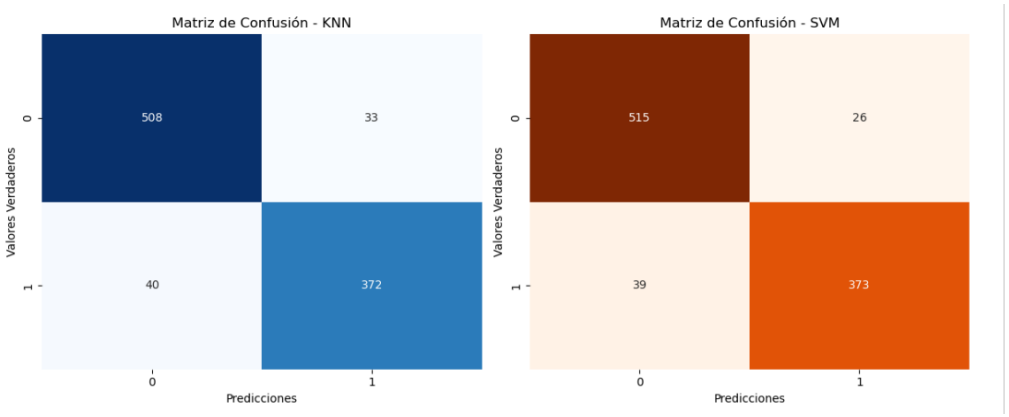


Según los resultados obtenidos, parece que el método SVM demuestra un mejor rendimiento y un desempeño más sólido en comparación con KNN. En esta etapa inicial de análisis, la evidencia sugiere que SVM es la elección preferida debido a su mayor eficacia en la clasificación de los datos. Pues esta tiene una mejor forma en contraste con la otra utilizando el criterio de la "diagonal del cuadrado".

Sin embargo, es importante recordar que la selección del método de aprendizaje automático adecuado depende de varios factores, como la naturaleza de los datos, los objetivos específicos del proyecto y las métricas de evaluación utilizadas. Por lo tanto, haremos más análisis y consideraremos otros factores antes de tomar una decisión final sobre el modelo a utilizar.

6.2. Comparación Matriz de Confusión

Ahora, veremos cómo se comportaron los modelos frente a la predicción del tipo de arroz.



Como se observa nuevamente en esta subsección, el método SVM muestra un desempeño superior en la tarea de detección de la clase de arroz en comparación con el método KNN, pues SVM tuvo un menor error en la elección de la clase 73 errores para el método KNN y 65 para el de SVM. Estos resultados refuerzan la idea de que SVM es una elección sólida y promisoria como el mejor método predictor para este problema particular.

Sin embargo, es importante subrayar que la elección del método óptimo puede depender de diversas consideraciones, y se debe realizar una buena evaluación antes de tomar una decisión definitiva. Factores específicos del problema también deben ser tomados en cuenta. Por lo tanto, aunque SVM se perfila como un fuerte candidato, es esencial llevar a cabo una última evaluación antes de tomar una decisión final sobre la elección del modelo predictivo.

6.3. Comparación de otras métricas asociadas

Ahora, para finalizar podemos medir el desempeño de cada uno de los métodos con distintas métricas asociadas vistas durante el curso. En efecto, note que:

```
Modelo: KNN
Accuracy: 0.9233997901364114
Precision: 0.9185185185185185
Recall: 0.9029126213592233
F1 Score: 0.9106487148102815

Modelo: SVM
Accuracy: 0.9317943336831059
Precision: 0.9348370927318296
Recall: 0.9053398058252428
F1 Score: 0.9198520345252774
```

De donde, podemos concluir finalmente que el método de SVM, es el que mejor se adapta y ha obtenido mejor desempeño. En todas las pruebas realizadas y en las comparaciones de este documento, ha logrado de forma sólida llevarse por delante al método de KNN.

7. Codigos Utilizados

7.1. Procesamiento datos y estadísticos

```
import pandas as pd
import matplotlib.pyplot as plt

path = "C:/Users/asus/Desktop/cosas_R/rice.txt"

df = pd.read_csv(path, sep=",", header=None, names=["Area", "Perimetro",
    "Longuitud eje mayor", "Lonjitud eje menor", "Excentricidad", "Area
    convexa", "Extension", "Especie"])

summary = df.describe()

print("Resumen Estadistico de las Caracteristicas Morfologicas:")
print(summary)

fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(16, 8))
fig.subplots_adjust(hspace=0.5)

features = ["Area", "Perimetro", "Longuitud eje mayor", "Lonjitud eje
    menor", "Excentricidad", "Area convexa", "Extension"]
for i, feature in enumerate(features):
    row = i // 4
    col = i % 4
    ax = axes[row, col]
    df[feature].plot(kind='hist', bins=20, ax=ax)
    ax.set_title(f'Histograma de {feature}')
    ax.set_xlabel(feature)

plt.show()
```

7.2. Histogramas asociados:

```
clases_arroz = df["Especie"].unique()

fig, axs = plt.subplots(nrows=len(df.columns[:-1]), ncols=1, figsize=(8, 12))
for i, caracteristica in enumerate(df.columns[:-1]):  # Excluye la columna
    "Especie"
    axs[i].set_title(f'Histograma de {caracteristica}')
    for clase in clases_arroz:
        subset = df[df["Especie"] == clase]
        axs[i].hist(subset[caracteristica], bins=20, alpha=0.5, label=clase)
    axs[i].set_xlabel(caracteristica)
    axs[i].set_ylabel('Frecuencia')
    axs[i].legend()

plt.tight_layout()
plt.show()
```

7.3. Gráficos de Dispersión y Matriz de Correlación

En este código, además se grafica la dispersión en un plano por separado para cada clase, pero esto no se agregó para no rellenar de datos el documento.

```
    # Separar el DataFrame en dos DataFrames segun la clase de arroz (Cammeo y Osmancik)
    df_cammeo = df[df["Especie"] == "Cammeo"]
    df_osmancik = df[df["Especie"] == "Osmancik"]

    correlation_matrix_general = df.corr()

    correlation_matrix_cammeo = df_cammeo.corr()
    correlation_matrix_osmancik = df_osmancik.corr()

    plt.figure(figsize=(10, 8))
    sns.heatmap(correlation_matrix_general, annot=True, cmap='coolwarm', linewidths=0.5)
    plt.title("Matriz de Correlacion para todo el Conjunto de Datos")
    plt.show()

    plt.figure(figsize=(10, 8))
    sns.heatmap(correlation_matrix_cammeo, annot=True, cmap='coolwarm', linewidths=0.5)
    plt.title("Matriz de Correlacion para la Clase Cammeo")
    plt.show()

    plt.figure(figsize=(10, 8))
    sns.heatmap(correlation_matrix_osmancik, annot=True, cmap='coolwarm', linewidths=0.5)
    plt.title("Matriz de Correlacion para la Clase Osmancik")
    plt.show()

    sns.pairplot(df, hue='Especie', markers=["o", "s"])
    plt.suptitle("Graficos de Dispersion entre Pares de Caracteristicas para todo el Conjunto de Datos", y=1.02)
    plt.show()

    sns.pairplot(df_cammeo, hue='Especie', markers=["o"])
    plt.suptitle("Graficos de Dispersion entre Pares de Caracteristicas para la Clase Cammeo", y=1.02)
    plt.show()

    sns.pairplot(df_osmancik, hue='Especie', markers=["s"])
    plt.suptitle("Graficos de Dispersion entre Pares de Caracteristicas para la Clase Osmancik", y=1.02)
    plt.show()
```

7.4. Preparación de los datos

Los códigos de esta sección se encuentran en el mismo documento en la misma sección. Por lo que no se colocaron nuevamente aquí.

7.5. Método KNN

```
    model_base_knn = KNeighborsClassifier(algorithm='ball_tree')
    param_grid_knn = {
        'n_neighbors' : [1, 3, 5, 8, 10, 12, 15, 30, 100]
    }
    best_knn = GridSearchCV(model_base_knn,
                            param_grid_knn,
                            cv = 5, # cantidad de divisiones al train set
                            scoring = 'f1' # 'accuracy', 'precision', 'recall',
                                'f1'
                            )
    best_knn.fit(X_train, y_train)

    y_pred_knn = best_knn.predict(X_test)

    ConfusionMatrixDisplay.from_predictions(
        y_test,
        y_pred_knn,
        cmap= 'Reds',
        display_labels = ["Osmancik","Cammeo"] # nombres de etiquetas
    );

    best_knn.best_params_

    RocCurveDisplay.from_estimator(
        best_knn,
        X_test,
        y_test,
        name = 'KNN',
        lw=3, color='red', zorder=2
    )
    plt.plot([0,1], [0,1], c='navy', ls='--', lw=3, zorder=1)
    plt.xlabel('Especificidad'); plt.ylabel('Sensibilidad')
    plt.title('Curva ROC: KNN');

    print(classification_report(y_test, y_pred_knn, target_names=["Cammeo",
       "Osmancik"]))

    f1_score_knn = f1_score(y_test, y_pred_knn)
    f1_score_knn
```

7.6. Método SVM

```
    model_base_svm = SVC()
    param_grid_svm = {
        'C' : [0.1, 1, 10],
        'kernel' : ['linear', 'rbf', 'poly'],
        'degree' : [2, 3],
    }
    best_svm = GridSearchCV(model_base_svm,
                            param_grid_svm,
                            cv = 5,
                            scoring = 'f1'
                            )
    best_svm.fit(X_train, y_train)

    best_svm.best_params_

    y_pred_svm = best_svm.predict(X_test)

    f1_score_svm = f1_score(y_test, y_pred_svm)
    f1_score_svm

    ConfusionMatrixDisplay.from_predictions(
        y_test,
        y_pred_svm,
        cmap= 'Greens',
        display_labels = ["Osmancik","Cammeo"] # nombres de etiquetas
    );

    RocCurveDisplay.from_estimator(
        best_svm,
        X_test,
        y_test,
        name = 'SVM',
        lw=3, color='green', zorder=2
    )
    plt.plot([0,1], [0,1], c='navy', ls='--', lw=3, zorder=1)
    plt.xlabel('Especificidad'); plt.ylabel('Sensibilidad')
    plt.title('Curva ROC: SVM');

    print(classification_report(y_test, y_pred_svm, target_names=["Cammeo",
       "Osmancik"]))
```

7.7. Comparación de las curvas ROC

```
    from sklearn.metrics import RocCurveDisplay

    # Configurar el grafico de la curva ROC
    plt.figure(figsize=(8, 6))
    roc_display_svm.plot(ax=plt.gca())  # Agregar la curva ROC de SVM al grafico actual
    roc_display_knn.plot(ax=plt.gca())  # Agregar la curva ROC de KNN al grafico actual

    # Linea de referencia (en este caso diagonal)
    plt.plot([0, 1], [0, 1], c='navy', ls='--', lw=3, zorder=1)

    plt.xlabel('Especificidad')
    plt.ylabel('Sensibilidad')
    plt.title('Curvas ROC: SVM vs. KNN')

    plt.legend(loc='lower right')
    plt.show()
```

7.8. Comparación valores finales

```
    accuracy_knn = accuracy_score(y_test, y_pred_knn)
    precision_knn = precision_score(y_test, y_pred_knn)
    recall_knn = recall_score(y_test, y_pred_knn)
    f1_score_knn = f1_score(y_test, y_pred_knn)

    accuracy_svm = accuracy_score(y_test, y_pred_svm)
    precision_svm = precision_score(y_test, y_pred_svm)
    recall_svm = recall_score(y_test, y_pred_svm)
    f1_score_svm = f1_score(y_test, y_pred_svm)

    metrics = {
        'KNN': {'Accuracy': accuracy_knn, 'Precision': precision_knn, 'Recall': recall_knn, 'F1 Score': f1_score_knn},
        'SVM': {'Accuracy': accuracy_svm, 'Precision': precision_svm, 'Recall': recall_svm, 'F1 Score': f1_score_svm}
    }

    for model, scores in metrics.items():
        print(f'Modelo: {model}')
        for metric, score in scores.items():
            print(f'{metric}: {score}')
        print()
```