Proyecto 2

Proyecto de clasificación de señas de manos

1st Sergio Sotil Lozada
Estudiante de la carrera de Computer Science
UTE
201810603
sergio.sotil@utec.edu.pe

2rd Mario Jacobo Rios Gamboa
Estudiante de la carrera de Computer Science
UTEC
201910285
mario.rios@utec.edu.pe

***Resumen*—En el presente informe se discutirán los resultados obtenidos luego de clasificar un set de señas de mano a través de los algoritmos: K-Nearest Neighbors, Support Vector Machine y Decision Tree.**

***Términos de referencia*—clasificación,seña de manos, algoritmo**

I. Introducción

La clasificación es un problema de aprendizaje supervisado que consiste en asignar una etiqueta a un conjunto de datos. En este caso, se tiene un set de imágenes de señas de manos, las cuales se clasificarán en 24 categorías mediante K-Nearest Neighbors, Support Vector Machine y Decision Tree. Finalmente, también se validará mediante indicadores cuantitativos y gráficas el desempeño de las técnicas utilizadas.

II. Explicación de modelos

II-A. K-Nearest Neighbors

Este algoritmo consiste en clasificar un nuevo dato basándose en los K datos más cercanos a él. Para facilitar este proceso, se emplea una estructura de datos que permita almacenar datos con múltiple dimensionalidad. Una de las estructuras de datos más utilizadas para este propósito es el árbol KD 1, el cual permite almacenar datos de múltiples dimensiones de forma eficiente empleando los mismos principios que un árbol binario.

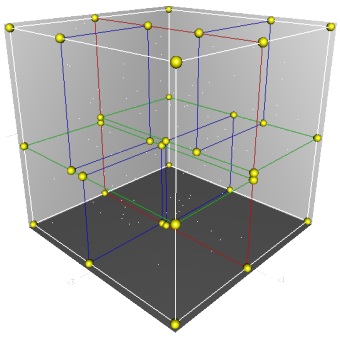

Figura 1: Árbol KD

De esta forma, se calculan los K vecimos más cercanos y se predice la clase de acuerdo a la clase predominante en los vecinos.

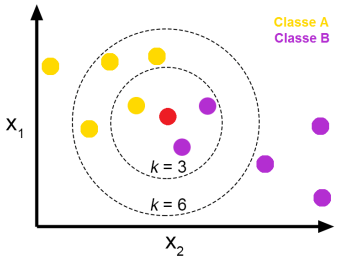

Figura 2: Ejemplo de clasificación con K-Nearest Neighbors

La métrica ”distancia” puede ser calculada de distintas formas. Entre ellas tenemos a la distancia de Manhattan, la cual se calcula sumando las diferencias absolutas de las coordenadas de los puntos. Otra métrica es la distancia Euclidiana, la cual se calcula sumando las diferencias al cuadrado de las coordenadas de los puntos. Finalmente, de forma general se tiene la distancia de Minkowski, la cual se calcula elevando a una potencia la suma de las diferencias de las coordenadas de los puntos.

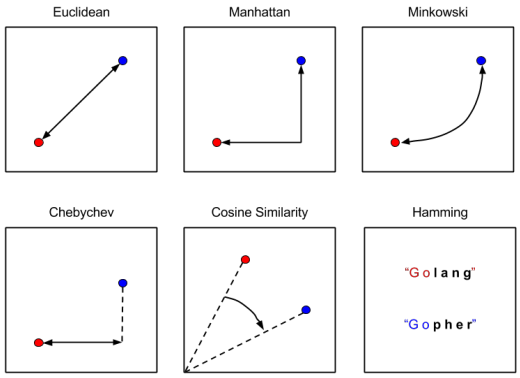

Figura 3: Tipos de distancia

En términos de modelado, en una clasificación KNN podemos encontrar los siguientes hiperparámetros:

- K: número de vecinos más cercanos a considerar

- Tipo de distancia: distancia de Manhattan, distancia Euclidiana o distancia de Minkowski
- Estructura de datos: árbol KD, árbol R, árbol R de Hilbert, entre otros

II-B. Support Vector Machine

Este algoritmo consiste en encontrar un hiperplano que maximice la distancia entre los puntos más cercanos de cada clase. De esta forma, se puede clasificar un nuevo dato asignándole la clase del lado en el que se encuentre.

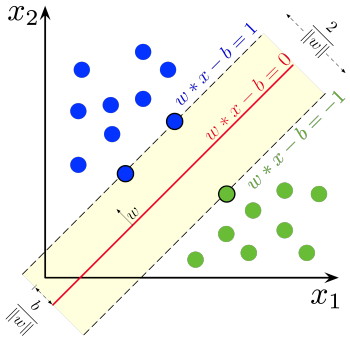

Figura 4: Ejemplo de clasificación con Support Vector Machine

Mientras más largo es el margen del vector de soporte, se reduce potencialmente, el error de generalización del clasificador. A diferencia de KNN, SVM no requiere de una estructura de datos para almacenar los datos de entrenamiento.

Este algoritmo emplea un kernel para procesar y transformar la data de entrada. Podemos encontrar kernels como:

- Kernel Gaussiano
- Kernel Polinomial
- Kernel RBF (Radial Basis Function)
- Kernel Sigmoidal

Finalmente, el kernel vendría a ser el principal hiperparámetro que este método emplea.

II-C. Decision Tree

Este algoritmo de clasificación emplea la estructura de un árbol. Los nodos internos representan las características del dataset, las ramas representan las reglas de decisión y cada nodo hoja representa el resultado. Para entrenar este modelo, se requiere construir el árbol escogiendo en cada nivel una características para partir la información. Para decidir la mejor característica, se emplea una métrica de impureza. Las métricas más comunes son:

- Ganancia de información
- Entropía
- Gini

En caso se tenga una característica continua y numérica se deberá emplear algún criterio para partir los datos, los más comunes son:

- Promedio
- Mediana

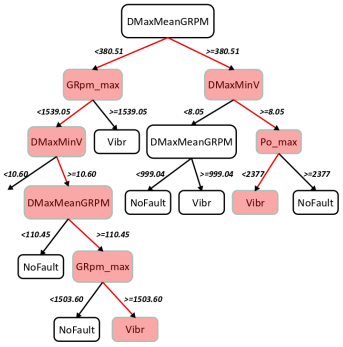

Figura 5: Ejemplo de clasificación con Decision Tree

Finalmente, para el decision tree podemos encontrar los siguientes hiperparámetros:

- Criterio de partición de la data
- Métrica de impureza

III. Configuración de entorno y de modelos

Para todo el proyecto, se empleó el cluster especializado de UTEC - Khipu. El mismo que cuenta con la siguiente configuración:

RAM	N Procesadores	Arquitectura	Sistema operativo
126 GB	80	x86_64	CentOS Linux 7

De los 3 clasificadores propuestos, el Decision Tree y el KNN han sido implementados utilizando el lenguaje de programación C++. En el Decision Tree se utilizó paralelismo compartido utilizando la librería OpenMP 6. De esta forma, se reducía de manera considerable el tiempo de entrenamiento. En efecto, se logró reducir el tiempo de **2 horas a 10 minutos** para el dataset entero de **34627 datos y 784 dimensiones**.

```
    std::vector<std::string> disordered_features = std::vector<std::string>(
#pragma omp parallel for
    for (int i = 0; i < features_casted.size(); i++) {
        auto feature = features_casted[i];

        auto label_values = get_unique<int>(df_categorical.get_header_column
        auto total_data = df_categorical.get_nrows();

        auto entropy = get_entropy(feature, df_categorical, label_values, to
        feature_entropy[i] = entropy;
        disordered_features[i] = feature;
    }
    auto min = std::min_element(feature_entropy.begin(), feature_entropy.end
    auto min_index = std::distance(feature_entropy.begin(), min);
    return disordered_features[min_index];
```

Figura 6: Paralelismo compartido aplicado a el cálculo de la entropía para cada característica

En el caso del SVM, se empleó la librería de Python Scikit-learn para entrenar el modelo. A continuación se detalla más información de la configuración de hiperparámetros de cada modelo:

- KDTree
  - Distancia: Euclidiana
  - K: Variable entre 1 y 4 (métricas aparte del ROC consideran el caso K = 4)
- SVM
  - Kernel: RBF

- Threshold: Variable dado por la librería

- Decision Tree
  - Medida de impureza: Entropía
  - Criterio de partición: Variable entre media y mediana (métricas aparte del ROC consideran la media)

Se emplearán los hiperparámetros variables para generar las curvas ROC.
Debido al gran poder de cómputo, no se vio necesario realizar algún tipo de reducción de dimensionalidad, pues se podía emplear la data completa en el entrenamiento de los modelos.

IV. Procesamiento de datos

La base de datos otorgada contaba con **34627 datos, cada uno de 784 dimensiones (pixeles)**. La etiqueta de cada dato estaba asignado a la columna 'label' y contaba con un total de 24 valores diferentes.

```
>>> df['label'].nunique()
24
>>>
```

Figura 7: Distribución de las etiquetas en la base de datos

```
>>> df.describe()
              label        pixel1  ...      pixel783     pixel784
count  34627.000000  34627.000000  ...  34627.000000  34627.00000
mean      11.473388    145.857077  ...    160.211309    158.77919
std        6.922215     41.839313  ...     64.395504     65.25230
min        0.000000      0.000000  ...      0.000000      0.00000
25%        5.000000    122.000000  ...    124.000000    121.00000
50%       11.000000    151.000000  ...    182.000000    181.00000
75%       17.000000    175.000000  ...    205.000000    204.00000
max       23.000000    255.000000  ...    255.000000    255.00000

[8 rows x 785 columns]
```

Figura 8: información del dataset

Primeramente, se notó que en el dataset, la relación de clases estaba desfasada (la clase 9 no existía) así que se corrigió este error reordenando los valores y sustituyendo los defectuosos. Seguidamente, se cargó toda la información en memoria y se emplearon 2 técnicas para partir la información y calcular las métricas.

El primer método, KFold Cross Validation, permite dividir el dataset entero K veces (folds), cada fold tendrá un segmento de datos de training y otros de validación. Esta división se da sin repetición.

El segundo método llamado Bootstraping, realiza un sampling de la información de manera análoga al KFold, con la diferencia de que este tiene repetición.

En ambos casos, se partió la data entera en 3 folds.

V. Resultados

Se emplearon 4 métricas para evaluar el rendimiento de los modelos:

- Precisión
- Recall
- F1 Score
- ROC AUC

Para el caso de la métrica ROC AUC, se emplearon los hiperparámetros variables para obtener los umbrales. En el caso de los otros 3 indicadores, se utilizaron las condiciones expuestas con anterioridad.
Cabe resaltar que como este es un problema de clasificación de múltiples clases, entonces para obtener el valor de los indicadores se siguió una estrategia OVR (One vs Rest) lo que implica que cada característica se evaluó de forma binaria como "Pertenece a la característica" o "No pertenece a la característica (resto)".
Finalmente, se obtuvo el promedio balanceado (weighted) de los valores obtenidos en cada característica.

A continuación se muestran los resultados de las métricas para cada modelo utilizando KFold Cross Validation y Bootstraping:

Modelo	Precisión	Recall	F1 score	AUC ROC
SVM	0.968546	0.966672	0.966861	0.999084
KNN	0.326510	0.294539	0.299693	0.633718
Decision Tree	0.775619	0.773960	0.773440	0.861332

Cuadro I: Resultados de las métricas para cada modelo utilizando KFold Cross Validation - K = 3

Modelo	Precisión	Recall	F1 score	AUC ROC
SVM	0.999765	0.999765	0.999765	1.000000
KNN	0.359328	0.323336	0.330016	0.648252
Decision Tree	0.856045	0.856045	0.856161	0.902447

Cuadro II: Resultados de las métricas para cada modelo utilizando Bootstraping - 3 folds

Se presentan también los mismos resultados a forma de gráfico de barras:

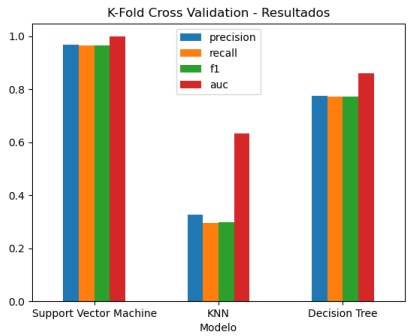

Figura 9: Resultados de las métricas para cada modelo utilizando KFold Cross Validation - K = 3

Podemos observar que el modelo SVM obtiene los mejores resultados en todas las métricas, seguido por el Decision Tree y finalmente el KNN.

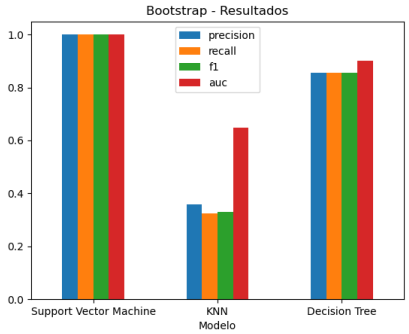

Figura 10: Resultados de las métricas para cada modelo utilizando Bootstraping - 3 folds

Una mayor precisión en el modelo SVM implica que dentro de los datos etiquetados como verdaderos. La mayor cantidad será efectivamente, verdadera. Este atributo es particularmente necesario en la clasificación de señas, pues mostrarle al usuario una seña incorrecta puede potencialmente alterar el significado de lo que se trataba de expresar.

Por otra parte, el modelo SVM también obtuvo los mejores resultados en cuanto a recall. Esto significa que de la cantidad de datos que son verdaderos, el modelo SVM es capaz de identificar la mayor cantidad de ellos.

El valor del F1 score es una combinación de precisión y recall, por lo que es una buena métrica para comparar estos dos indicadores de forma conjunta entre los métodos. No obstante, debido a que el problema consiste en la identificación de señas, la precisión es más importante que el recall.

La curva ROC nos proporciona una vista gráfica de la relación entre el ratio de falsos positivos con el ratio de verdaderos positivos. Si el modelo tiene poder predictivo, entonces estará encima de la línea diagonal que pasa por el origen.

A continuación se muestran las curvas ROC para cada modelo en el primer fold de la división KFold Cross Validation:

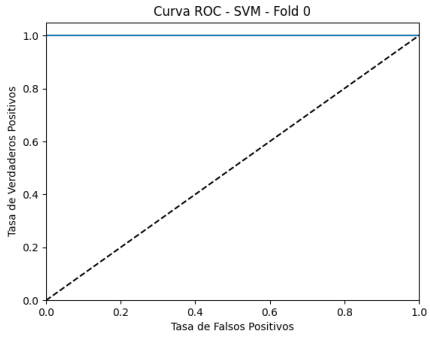

Figura 11: Curva ROC para el modelo SVM en el tercer fold de la división KFold Cross Validation

De las gráficas en conjunto con el valor AUC de las curvas (área) y los indicadores anteriores, concluimos que el clasificador SVM es el mejor modelo para este problema

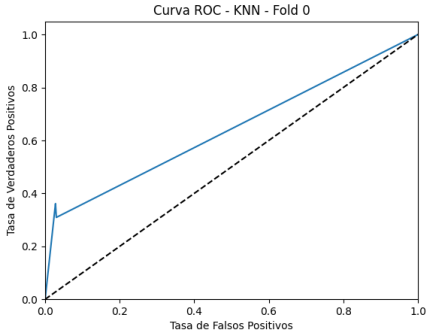

Figura 12: Curva ROC para el modelo KNN en el tercer fold de la división KFold Cross Validation

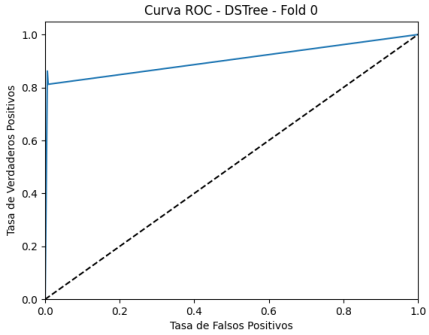

Figura 13: Curva ROC para el modelo Decision Tree en el tercer fold de la división KFold Cross Validation

llegando a comportarse de forma similar a un clasificador perfecto.

El árbol de decisión posee también una fuerte capacidad predictiva. En cuanto al método KNN, aunque las primeras 3 métricas no son significativas, la curva ROC nos indica que tiene cierto poder predictivo y es mejor que un clasificador aleatorio.

VI. Conclusiones

Se concluye que, el modelo SVM es el mejor para este problema, seguido por el Decision Tree y el KNN. Para futuras investigaciones, se podría probar variar más hiperparámetros en el Decision Tree como por ejemplo, el criterio para medir la impureza. Así como el tipo de distancia y la estructura usada en el KNN. Estructuras como el RTree o el Hilbert RTree permiten indexar de forma más eficiente puntos y objetos multidimensionales facilitando así la búsqueda de los vecinos más cercanos.

A lo largo del proyecto se ha comprobado como diversos clasificadores se pueden aplicar a un problema de clasificación de señas. De la misma forma, como distintas técnicas de evaluación de modelos (muestreos) y métricas explican distintas características que pueden ser de diferente relevancia de acuerdo a la naturaleza del problema.

REFERENCIAS

[1] D. P. Mehta, S. Sahni, “Handbook of data structures and applications”, CRC Press, 2018.

[2] N. Matloff, “Statistical Regression and Classification: From Linear Models to Machine Learning”, Chapman and Hall/CRC, 2017.