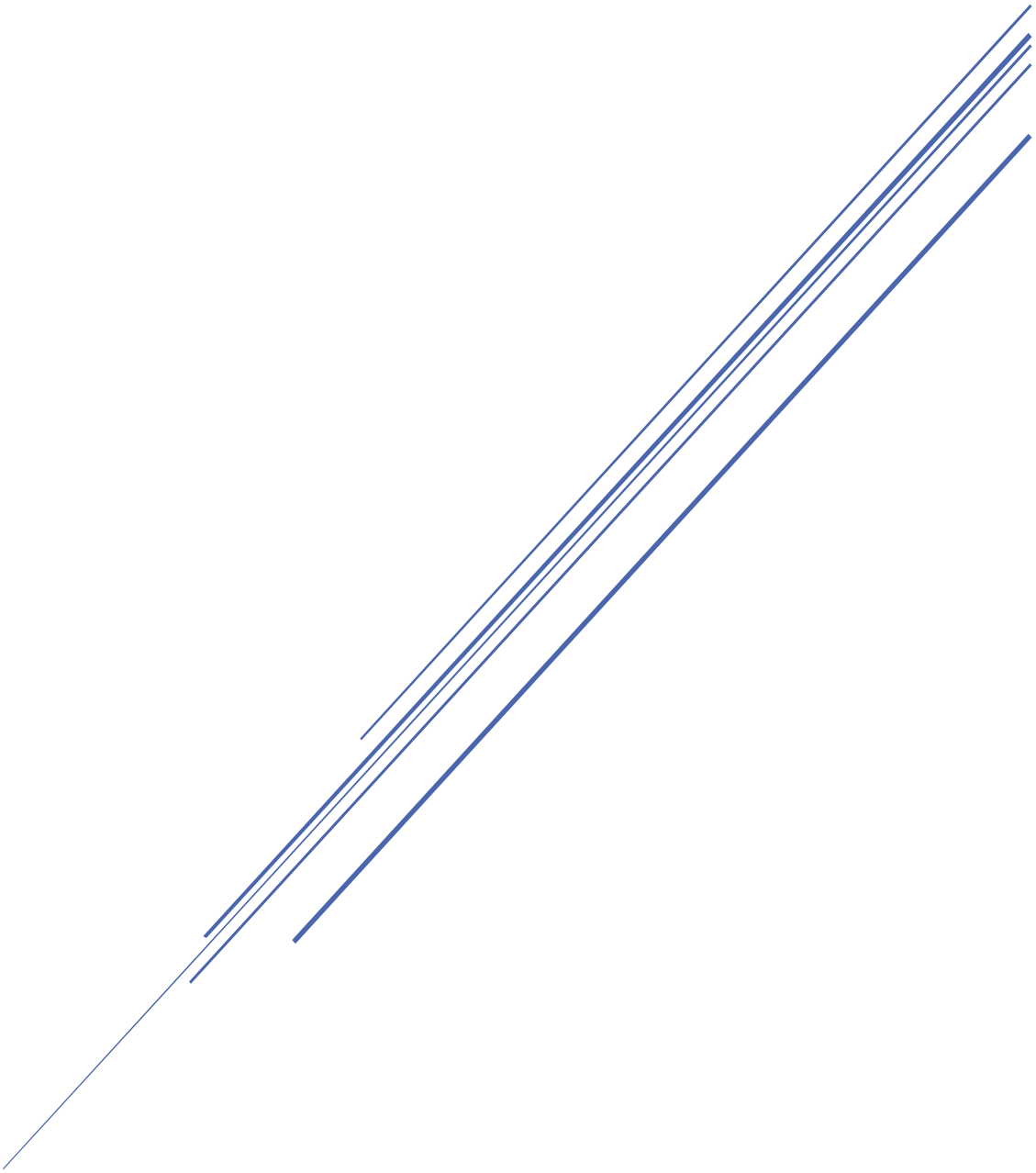
PROYECTO FINAL

Inteligencia Artificial I



Galdós, Ignacio
11036

Índice

Introducción

El presente proyecto tiene por objetivo obtener a partir de la imagen de un auto, las letras y números de su patente. Para realizarlo se optó por el análisis de cuatro algoritmos para encontrar el que mejor resuelva el problema.

Se decidió utilizar el formato de patente argentina de 1994, dado que presenta un formato más sencillo para la obtención del dataset. A pesar de esto, los resultados se pueden extrapolar hacia el formato de patente del Mercosur introducido en 2015, realizando unas pequeñas modificaciones.

Como el tipo de problema dificulta la obtención del dataset, el utilizado finalmente es reducido a solo 110 elementos. Por esta razón se aplicaron diferentes enfoques para poder resolver el problema. La solución final mejoraría y se podrían usar otros tipos de enfoques si se incrementara el numero de casos del dataset de entrenamiento.

Se decidió utilizar el entorno de desarrollo de Google Colab, que implica el uso del lenguaje Python, que resulta ser el más adecuado para este caso. También se decidió utilizar la librería de inteligencia artificial llamada Sklearn, ya que esta provee una cantidad de algoritmos y métricas para medir su rendimiento.

En este informe se explica en primer lugar el funcionamiento de los algoritmos seleccionados, luego se detalla la obtención y tratamiento del dataset. Se presenta un análisis detallado de los resultados de cada algoritmo variando sus parámetros. Luego se detalla el algoritmo a utilizar con sus ajustes de parámetros.

Finalmente se presenta una conclusión del proyecto junto a las posibles mejoras y posibilidades de continuidad.

Todos los archivos mencionados y el dataset correspondientes son adjuntados junto con este informe.

Marco teórico

Se decidió evaluar y contrastar los siguientes algoritmos. Los mismos fueron tomados con su variante de clasificación ya que se presenta un problema de este tipo.

KNeighbors

La idea clave de KNeighbors es que: es probable que las propiedades de un punto de entrada particular x sean similares a las de los puntos cercanos a x.

En este proyecto se usa su variante para clasificación cuya salida es la pertenencia a una clase. Un objeto es clasificado como perteneciente a una clase si la mayoría de sus k vecinos o vecindad pertenecen a esa clase. Si se quiere clasificar un objeto, con k= 3 vecinos, con 3 vecinos de la clase A y 2 de la clase B, por mayoría el objeto se clasifica como clase A. Para evitar empates, se prefiere que el número de vecinos k seleccionado sea un número impar.

El reto que presenta este algoritmo es definir el tamaño de la vecindad o valor de k. Si es muy pequeña, no contendrá ningún punto de los datos; si es muy grande, puede incluir a todos los puntos de los datos, dando lugar a una estimación que es la misma en cualquier lugar. Una solución es definir la vecindad lo suficientemente grande como para incluir k puntos, donde k es lo suficientemente grande como para asegurar una estimación con significado. Si los datos están dispersos, la vecindad es grande, pero si los datos están muy juntos, la vecindad es pequeña.

Decision Tree

Un árbol de decisión toma como entrada un objeto o una situación descrita a través de un conjunto de atributos y devuelve una "decisión" (el valor previsto de la salida dada la entrada). Los atributos de entrada pueden ser discretos o continuos, en el caso de este proyecto se usan datos discretos entre 0-255. El valor de la salida puede ser a su vez discreto o continuo; una función de valores discretos se denomina clasificación; una función continua se denomina regresión. En el caso de este informe se utilizará una salida discreta ya que se trata de resolver un problema de clasificación.

Un árbol de decisión desarrolla una secuencia de test para poder alcanzar una decisión. Cada nodo interno del árbol corresponde con un test sobre el valor de una de las propiedades, y las ramas que salen del nodo están etiquetadas con los posibles valores de dicha propiedad. Cada nodo hoja del árbol representa el valor que ha de ser devuelto si dicho nodo hoja es alcanzado. La representación en forma de árboles de decisión facilita el entendimiento de su funcionamiento interno.

Random Forest

Random Forest es un método versátil de machine learning capaz de realizar tanto tareas de regresión como de clasificación. Es un tipo de método de aprendizaje por conjuntos, donde un grupo de modelos débiles se combinan para formar un modelo poderoso.

Este algoritmo ejecuta varios algoritmos de árbol de decisiones en lugar de uno solo. Para clasificar un nuevo objeto basado en atributos, cada árbol de decisión da una clasificación y finalmente la decisión con mayor "votos" es la predicción del algoritmo.

Esto permite mejorar la estimación del resultado final además de poder manejar grandes cantidades de datos con mayor dimensionalidad. Puede manejar miles de variables de entrada e identificar las variables más significativas, por lo que se considera uno de los métodos de reducción de dimensionalidad. Tiene un método efectivo para estimar datos faltantes y mantiene la precisión cuando falta una gran proporción de los datos.

Support Vector Machines

Las máquinas de vectores de soporte son una técnica de machine learning que encuentra la mejor separación posible entre clases. Normalmente, los problemas de aprendizaje automático tienen muchísimas dimensiones. Así que, en vez de encontrar la línea óptima, el SVM encuentra el hiperplano que maximiza el margen de separación entre clases.

Esto permite encontrar la forma óptima de clasificar entre varias clases. La clasificación óptima se realiza maximizando el margen de separación entre las clases. Los vectores que definen el borde de esta separación son los vectores de soporte. En el caso de que las clases no sean linealmente separables, se puede usar otros kernels. En el caso de este informe se utilizan los siguientes Kernel:

- Lineal: en el que espacio se divide con líneas rectas.
- Polinómica: en el que se divide con diferentes polinomios en las que se puede ajustar su grado de inclinación. En el caso de que su grado sea 0 seria lineal.
- RBF: es una función de base radial cuyo valor depende sólo de la distancia del origen.

Se puede observar un ejemplo en la siguiente imagen.

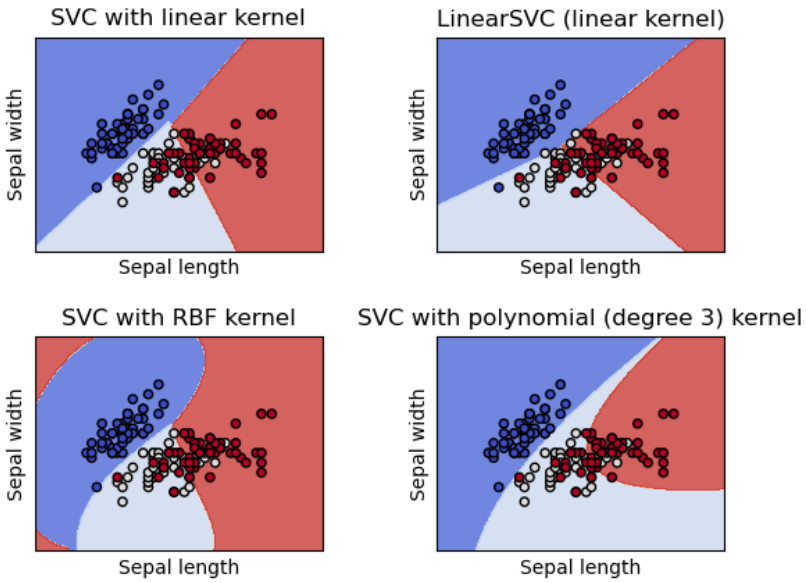


Imagen 1: kernel sklearn

Diseño experimental

Datasets

Se describen los pasos a seguir para obtener el dataset final.

Paso 1

En primera instancia se conforma un dataset de 110 imágenes de vehículos. A estas imágenes se las trasforma a formato jpg y luego se coloca como nombre de archivo la patente. Un ejemplo sería ABC123.jpg. Utilizando LabelImg se etiqueta cada imagen creando un archivo .xml. Este archivo contiene las coordenadas de la patente y las dimensiones de la imagen expresada en pixeles.

Paso 2

Se toma la imagen junto con su .xml y luego se recorta a partir de las coordenadas cargadas en el xml. Una vez recortada, se estandariza su resolución a 400x108 pixeles. Esto se realiza para poder aplicarles los algoritmos, ya que los mismos necesitan un conjunto de datos de igual tamaño. Posteriormente se transforma la imagen a

monocromática, es decir a cada píxel se le asigna un valor entre 0-255 que representa una intensidad de gris. Esto se realiza para eliminar datos innecesarios, ya que los colores producen variaciones poco relevantes en el análisis de este problema y un significativo aumento del costo computacional. Se obtiene una imagen como se muestra en la Imagen 2.

Imagen 2: recorte de patente

Parte 3

A partir de aquí surgen tres enfoques a desarrollar

- Utilizar la imagen como se muestra en la Imagen 2.
- Dividir a la mitad separando en letras y en números
- Dividir por carácter
    - A su vez se pueden separar letras y números

Primer enfoque

Se descarta el uso del primer enfoque ya que al tener un dataset pequeño de solo 110 patentes no es suficiente para poder encontrar relaciones entre ellos. Además de necesitar un conjunto más grande se necesitaría de datos más heterogéneo ya que la cantidad de combinaciones posibles es muy grande de 17.576.000.

Segundo enfoque

Se descarta el uso del segundo enfoque, ya que a pesar de reducir significativamente la cantidad de combinaciones posibles y de separar el problema en dos partes, letras y números, la cantidad de combinaciones es alta. Para las letras es de 17.576 y para los números de 1.000. Resultando en un dataset no suficientemente grande para poder aplicar este enfoque, ya que seria de 110 letras de 17576 combinaciones posibles y de 110 números de 1000 combinaciones posibles.

Tercer enfoque

Se opta por la utilización de este enfoque ya que, al dividir cada carácter no solo se reducen significativamente la cantidad de combinaciones, sino que también se aumenta los datos de entrenamientos. Esto se produce porque cada patente tiene 3 letras y 3 números. Se opta por separar en letras y números, lo que reduce aún más las combinaciones y posibles errores en caracteres similares como en la O y el número 0. Esto se puede hacer ya que en las patentes siempre siguen el esquema de tres letras y tres números.

Resulta en un dataset final de 330 letras con 26 opciones posibles y de 330 números con 10 opciones posibles.

Cada una de estas imágenes se trasforma en una lista de pixeles y se agregan a otra lista resultando en dos dataset finales, uno de letras y otro de números, cada uno siendo una matriz de 330*(altura*ancho) siendo 330*6588 en la que cada elemento es un número entre 0-255.

El proceso completo se puede observar en el diagrama 1.

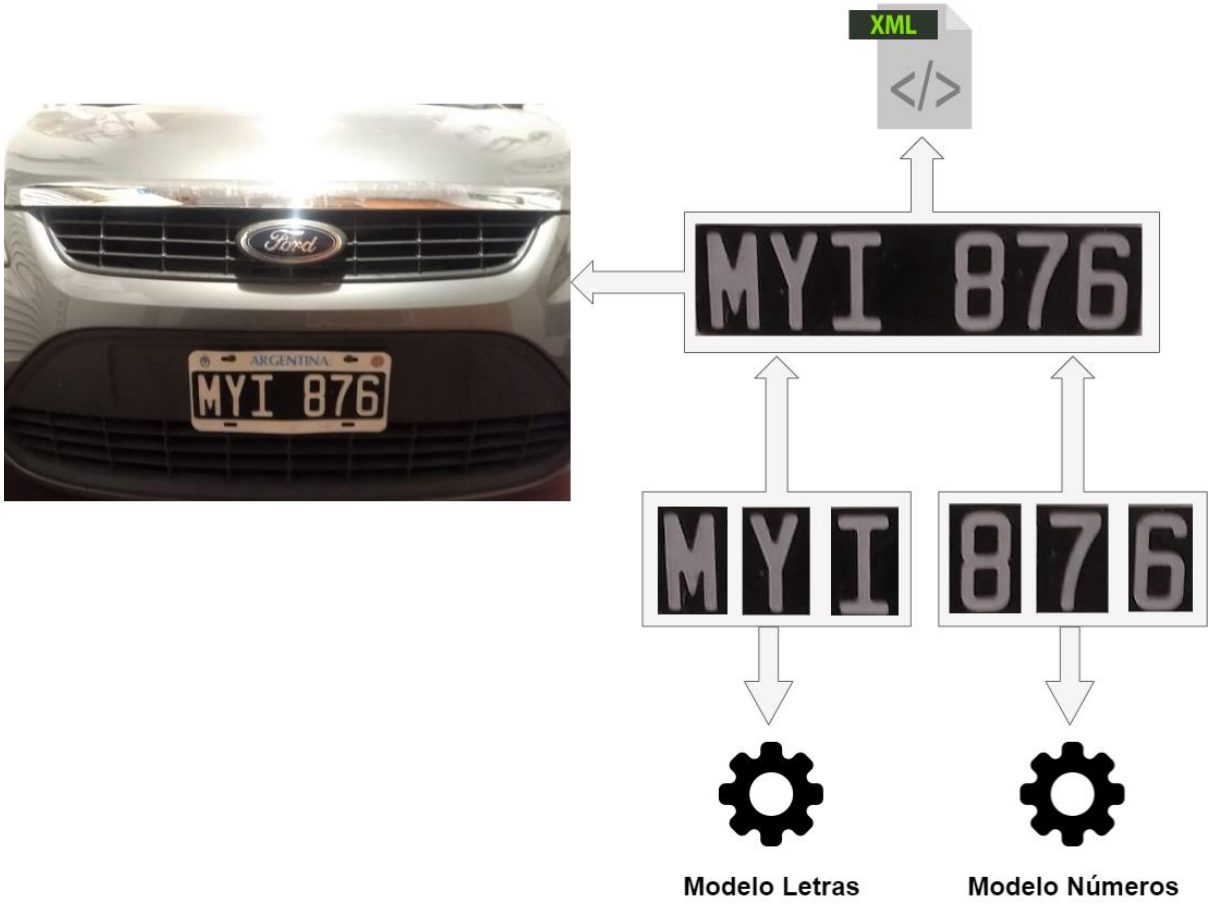


Diagrama 1: Tratamiento del dataset completo

Modelos

Primero se divide el dataset en entrenamiento y testeo. Esto se realiza para que una vez entrenado el modelo se pueda evaluar su rendimiento. El tamaño del conjunto de testeo elegido fue de 20% del total de dataset.

Una vez realizado esto se utiliza la función de *cross_val_predict* para evaluar los diferentes algoritmos con diferentes parámetros. Y se almacenan los resultados en un .csv. Esta función de la librería de sklearn funciona de la siguiente manera:

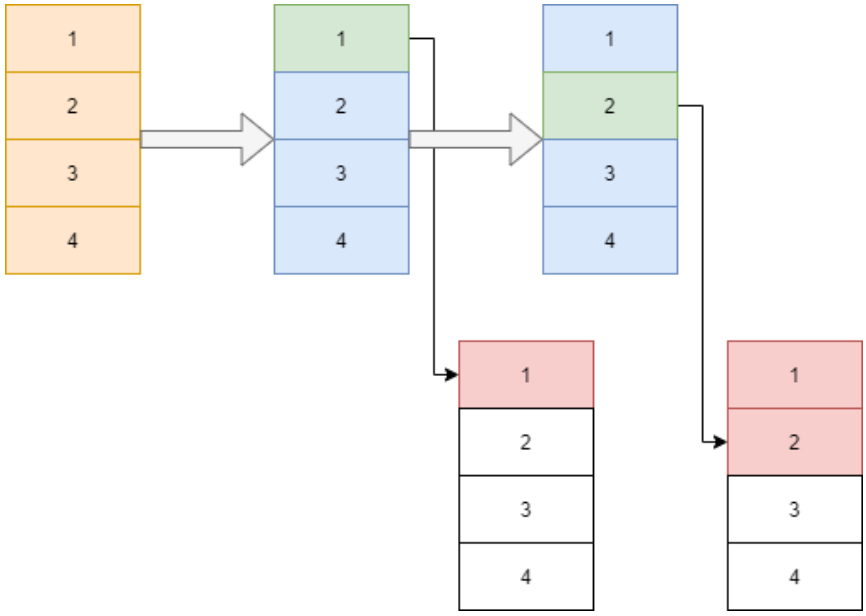


Diagrama 2: validacion cruzada de sklearn

Se decidió separar el conjunto de entrenamiento en cuatro partes como se observa en amarillo en el diagrama 2. Luego se realizan tres pasos

- Se toman 3 elementos para entrenar el modelo, en este caso el conjunto 2,3,4 marcados en azul
- y se evalúa con el elemento 1 marcado en verde
- para finalmente almacenar la predicción en una nuevo dataset marcado en rojo

Luego se entrena con el conjunto 1, 3 ,4 y se evalúa con el 2. Se continua así hasta evaluar cada uno de los conjuntos. Resultando en un nuevo dataset con la predicción de todos los elementos evaluados una vez.

Análisis y discusión de resultados

Se probaron diferentes algoritmos a partir del uso de validación cruzada, a fin de encontrar el óptimo para este problema. Después, se ensaya ejecutando variaciones en los parámetros del algoritmo seleccionado; para finalmente evaluar el conjunto de entrenamiento y de test.

Parte 1

En esta primera parte se contrastan los algoritmos para encontrar el de mejor solución al problema.

KNeighbors

Se evalua haciendo variar la cantidad de vecinos desde 1 a 20 , y se conforma un archivo llamado kneighbors_data.csv adjuntado al informe. Luego se crearon los siguientes gráficos teniendo como parámetros los vecinos y el porcentaje de aciertos.

Neighbors	Accuracy Leter	Accuracy Number
1	0,612121212	0,772727273
2	0,551515152	0,712121212
3	0,487878788	0,678787879
4	0,460606061	0,663636364
5	0,433333333	0,609090909
6	0,415151515	0,569696970
7	0,354545455	0,542424242
8	0,321212121	0,533333333
9	0,306060606	0,503030303
10	0,303030303	0,503030303
11	0,284848485	0,469696970
12	0,263636364	0,445454545
13	0,254545455	0,430303030
14	0,227272727	0,409090909
15	0,190909091	0,390909091
16	0,178787879	0,363636364
17	0,154545455	0,360606061
18	0,145454545	0,315151515
19	0,136363636	0,290909091

Tabla 1: accuracy neighbors

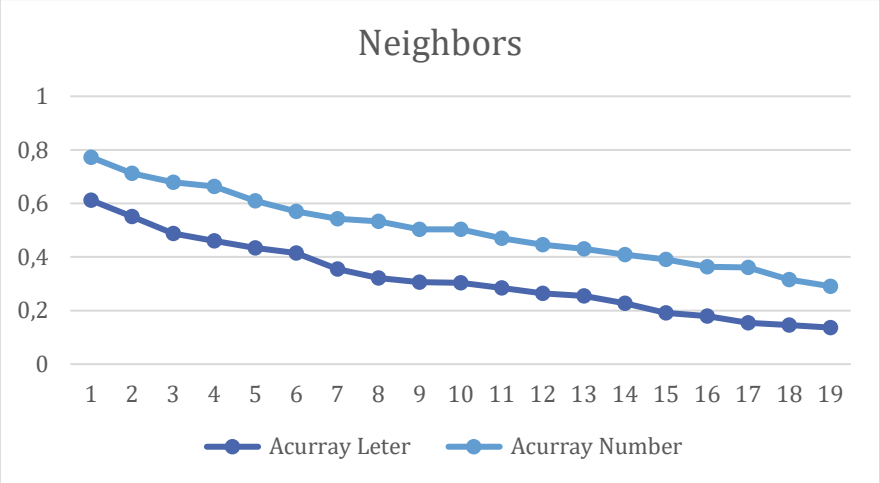


Diagrama 3: precisión a partir de cantidad de vecinos

Conclusiones preliminares

La variación que obtiene la mayor precisión es cuando solo se considera un vecino. A partir de esto se puede observar que los mejores parámetros para este algoritmo serian cercanos a:

- Letras: usar un vecino obteniendo una precisión aproximada de 61%
- Números: usar un vecino obteniendo una precisión aproximada de 77%

La ventaja de utilizar este algoritmo es que requiere poco tiempo de procesamiento en comparación a los demás.

Decision Tree

Se evaluó haciendo variar la profundidad del árbol desde 1-100. Con esto se creó un archivo llamado dt_data.csv. Luego se conformaron los siguientes gráficos.

max_depth	Accuracy Leter	Accuracy Number	max_depth	Accuracy Leter	Accuracy Number
1	0,106060606	0,209090909	100	0,600000000	0,663636364
2	0,190909091	0,369696970	200	0,593939394	0,657575758
3	0,348484848	0,575757576	300	0,600000000	0,678787879
4	0,503030303	0,651515152	400	0,593939394	0,672727273
5	0,600000000	0,663636364	500	0,636363636	0,669696970
6	0,600000000	0,681818182	600	0,600000000	0,681818182
7	0,618181818	0,669696970	700	0,575757576	0,669696970
8	0,584848485	0,654545455	800	0,578787879	0,678787879
9	0,609090909	0,672727273	900	0,630303030	0,672727273
10	0,606060606	0,696969697	1000	0,581818182	0,690909091

20	0,578787879	0,660606061	2000	0,618181818	0,675757576
30	0,590909091	0,684848485	3000	0,615151515	0,681818182
40	0,581818182	0,675757576	4000	0,615151515	0,651515152
50	0,590909091	0,660606061	5000	0,593939394	0,663636364
60	0,603030303	0,687878788	6000	0,603030303	0,675757576
70	0,593939394	0,669696970	7000	0,615151515	0,657575758
80	0,612121212	0,660606061	8000	0,600000000	0,712121212
90	0,600000000	0,672727273	9000	0,633333333	0,666666667

Tabla 2: decision tree

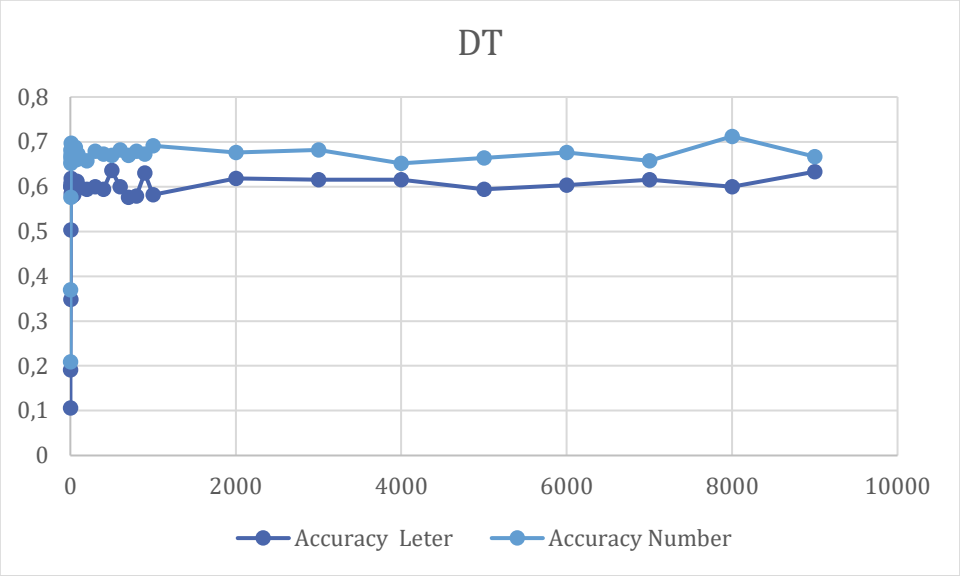


Diagrama 4: precisión en función de la profundidad

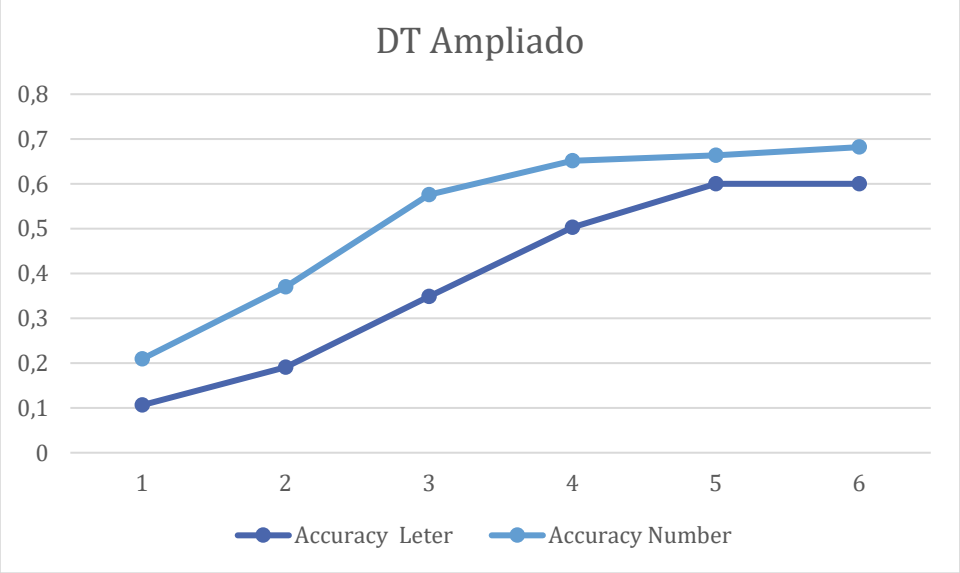


Diagrama 5: precisión en función de la profundidad detallado

Conclusiones preliminares

Como se observa en los diagramas, a partir de la profundidad del árbol 6 no hay cambios. Esto permite concluir que los mejores parámetros para este algoritmo rondarían:

- Letras: Usar 6 vecino obteniendo una precisión aproximada de 60%
- Números: Usar 6 vecino obteniendo una precisión aproximada de 68%

Random Forest

Se evalua haciendo variar la cantidad de árboles desde 1-500 y el criterio de división del árbol que provee Sklearn que son Gini y Entropy. Con esto se crea un archivo, llamado rf_data.csv. Se conforman los siguientes gráficos.

max_iter	Number Accuracy Entropy	Leter Accuracy Entropy	Number Accuracy Gini	Leter Accuracy Gini
1	0,609090909	0,472727273	0,624242424	0,487878788
2	0,581818182	0,496969697	0,596969697	0,439393939
3	0,675757576	0,548484848	0,687878788	0,557575758
4	0,757575758	0,639393939	0,742424242	0,578787879
5	0,760606061	0,630303030	0,763636364	0,636363636
6	0,787878788	0,639393939	0,784848485	0,633333333
7	0,784848485	0,721212121	0,766666667	0,663636364
8	0,790909091	0,721212121	0,796969697	0,678787879
9	0,818181818	0,703030303	0,800000000	0,718181818
10	0,818181818	0,727272727	0,803030303	0,687878788
20	0,854545455	0,784848485	0,866666667	0,721212121
30	0,851515152	0,793939394	0,824242424	0,748484848
40	0,848484848	0,787878788	0,857575758	0,772727273
50	0,875757576	0,784848485	0,860606061	0,806060606
60	0,851515152	0,809090909	0,863636364	0,793939394
70	0,878787879	0,803030303	0,878787879	0,781818182
80	0,872727273	0,800000000	0,869696970	0,787878788
90	0,872727273	0,818181818	0,869696970	0,790909091
100	0,860606061	0,806060606	0,866666667	0,784848485
200	0,866666667	0,809090909	0,878787879	0,800000000
300	0,878787879	0,812121212	0,872727273	0,800000000
400	0,878787879	0,818181818	0,872727273	0,787878788
500	0,857575758	0,803030303	0,872727273	0,787878788

Tabla 3: random forest

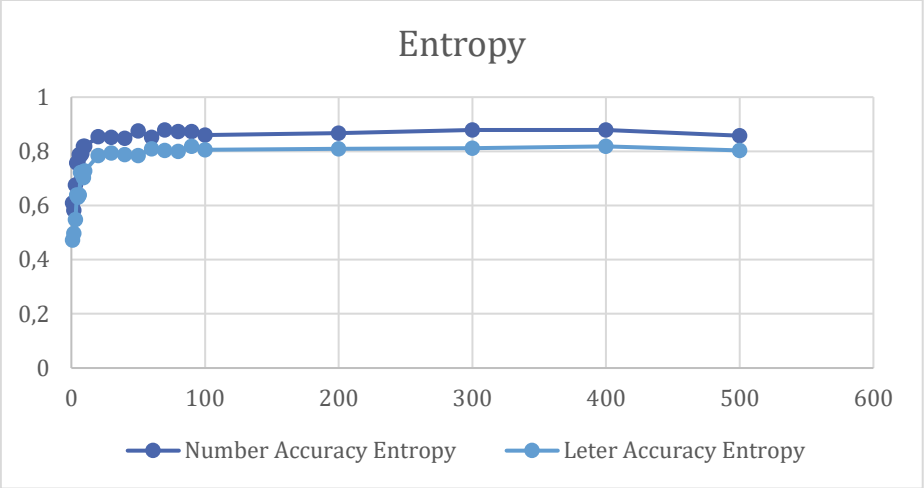


Diagrama 6: precisión a partir de cantidad iteraciones

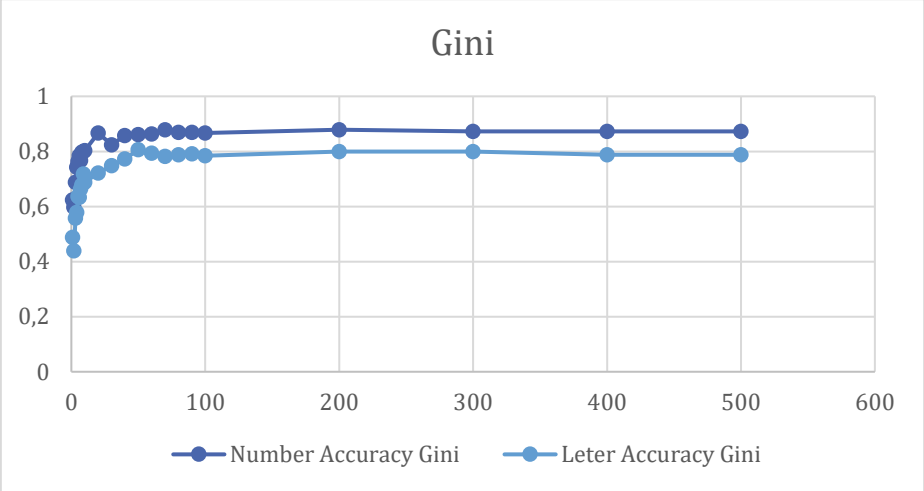


Diagrama 7: precisión a partir de cantidad iteraciones

Conclusiones preliminares

Como se puede observar en la Tabla 3, a partir de 60 iteraciones la precisión no mejora significativamente. Esto nos permite descartar ejecuciones innecesarias.

A partir de esto se puede observar los mejores parámetros para este algoritmo, que rondarían:

- Letras: usar método Entropy con una cantidad aproximada de 50 árboles. Obteniendo una precisión aproximada de 87%
- Números: usar método Entropy con una cantidad aproximada de 60 árboles. Obteniendo una precisión aproximada de 80%

Support vector machine

Se evalua haciendo variar la cantidad máxima de iteraciones desde 1-9000. Con esto se crea un archivo, llamado svm_data.csv. Luego se conforman las siguiente tabla y gráficos.

max_iter	Leter Accuracy ovr	Leter Accuracy crammer	Number Accuracy ovr	Number Accuracy crammer
1	0,460606061	0,915151515	0,657575758	0,936363636
2	0,603030303	0,915151515	0,800000000	0,936363636
3	0,715151515	0,915151515	0,842424242	0,936363636
4	0,727272727	0,915151515	0,872727273	0,936363636
5	0,769696970	0,915151515	0,881818182	0,936363636
6	0,809090909	0,915151515	0,900000000	0,936363636
7	0,827272727	0,915151515	0,890909091	0,936363636
8	0,790909091	0,915151515	0,933333333	0,936363636
9	0,854545455	0,915151515	0,912121212	0,936363636
10	0,836363636	0,915151515	0,933333333	0,936363636
20	0,903030303	0,915151515	0,933333333	0,936363636
30	0,909090909	0,915151515	0,951515152	0,936363636
40	0,900000000	0,915151515	0,954545455	0,936363636
50	0,893939394	0,915151515	0,948484848	0,936363636
60	0,887878788	0,915151515	0,945454545	0,936363636
70	0,890909091	0,915151515	0,945454545	0,936363636
80	0,887878788	0,915151515	0,945454545	0,936363636
90	0,890909091	0,915151515	0,945454545	0,936363636
100	0,884848485	0,915151515	0,942424242	0,936363636
200	0,887878788	0,915151515	0,942424242	0,936363636
300	0,878787879	0,915151515	0,942424242	0,936363636
400	0,884848485	0,915151515	0,942424242	0,936363636
500	0,881818182	0,915151515	0,942424242	0,936363636
600	0,881818182	0,915151515	0,942424242	0,936363636
700	0,884848485	0,915151515	0,942424242	0,936363636
800	0,884848485	0,915151515	0,942424242	0,936363636
900	0,884848485	0,915151515	0,942424242	0,936363636
1000	0,884848485	0,915151515	0,942424242	0,936363636
2000	0,884848485	0,915151515	0,942424242	0,936363636
3000	0,884848485	0,915151515	0,942424242	0,936363636
4000	0,884848485	0,915151515	0,942424242	0,936363636
5000	0,884848485	0,915151515	0,942424242	0,936363636
6000	0,884848485	0,915151515	0,942424242	0,936363636
7000	0,884848485	0,915151515	0,942424242	0,936363636
8000	0,884848485	0,915151515	0,942424242	0,936363636
9000	0,884848485	0,915151515	0,942424242	0,936363636

Tabla 4: precisión del algoritmo svm

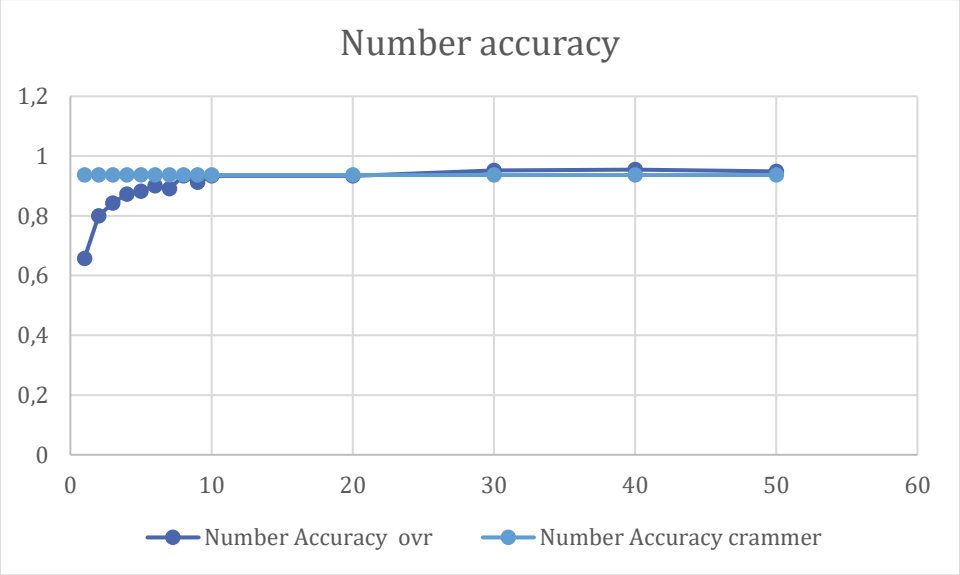


Diagrama 8: svm precisión en función de iteraciones

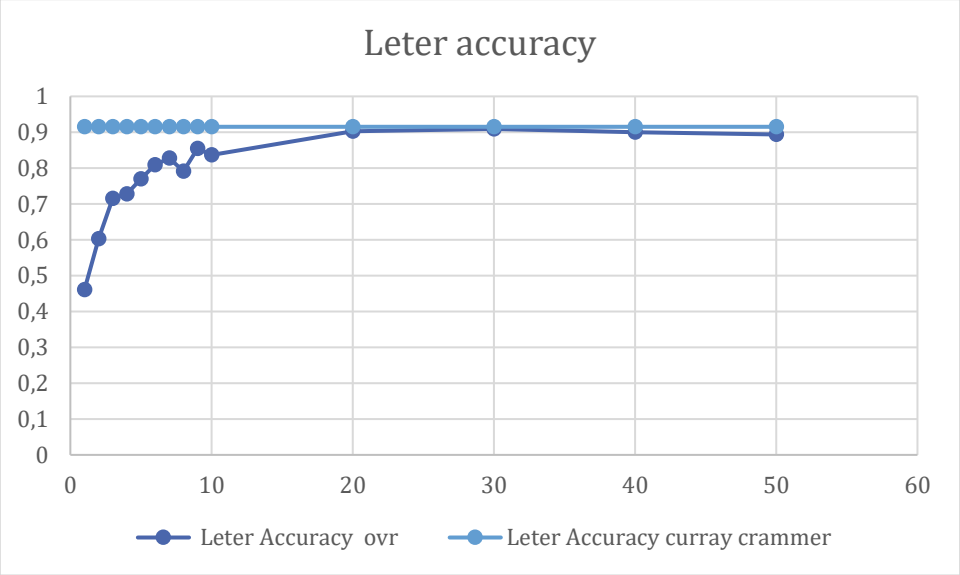


Diagrama 9: svm precisión en función de iteraciones

Conclusiones preliminares

A partir de observar la Tabla 2 y los diagramas, a partir de los mejores parámetros para este algoritmo rondaría:

- Letras: usar método Crammer Singercon una cantidad aproximada de 1 iteración. Obteniendo una precisión aproximada de 91%
- Números: usar el método OVR con una cantidad aproximada de 40. Obteniendo una precisión aproximada de 94%

Parte 2

En esta parte se explica como se optimizaron los parámetros del algoritmo elegido que es Suport vector machine.

Kernel

En primera instancia se elige probar otros tipos de Kernel aparte del lineal como el poly y rbf (Radial basis function)

max_iter	Number Accuracy Poly	Leter Accuracy Poly	Number Accuracy Rbf	Leter Accuracy Rbf	Number Accuracy Linear	Leter Accuracy Linear
1	0,848484848	0,706060606	0,121212121	0,06969697	0,96969697	0,909090909
2	0,848484848	0,706060606	0,121212121	0,06969697	0,96969697	0,909090909
3	0,848484848	0,706060606	0,121212121	0,06969697	0,96969697	0,909090909
4	0,848484848	0,706060606	0,121212121	0,06969697	0,96969697	0,909090909
5	0,848484848	0,706060606	0,121212121	0,06969697	0,96969697	0,909090909
6	0,848484848	0,706060606	0,121212121	0,06969697	0,96969697	0,909090909
7	0,848484848	0,706060606	0,121212121	0,06969697	0,96969697	0,909090909
8	0,848484848	0,706060606	0,121212121	0,06969697	0,96969697	0,909090909
9	0,848484848	0,706060606	0,121212121	0,06969697	0,96969697	0,909090909
10	0,848484848	0,706060606	0,121212121	0,06969697	0,96969697	0,909090909

Tabla 5: Kernel Accuracy

A partir de la Tabla 5 se puede concluir que el mejor Kernel para este caso es el lineal. Esto genera el desafío de probar la hipótesis de que disminuyendo el grado de inclinación del Kernel polinómico, proporcionaría un mejor resultado (ver tabla) No obstante, la mejor opción continuó siendo el Kernel lineal.

Grados	Number Accuracy	Leter Accuracy
1	0,969696970	0,912121212
2	0,921212121	0,809090909
3	0,848484848	0,703030303
4	0,796969697	0,621212121
5	0,736363636	0,560606061
6	0,690909091	0,521212121

Tabla 6: grados kernel polinómica

Tolerancia

Es el parámetro que indica la tolerancia al criterio de parada.

Se decide probar con más detalles variándola desde 0.0000001 a 10.0 y se conforma la siguiente tabla.

Tolerancia	Number Accuracy	Leter Accuracy
0.00001	0,912121212	0,969696970
0.0001	0,912121212	0,969696970
0.001	0,912121212	0,972727273
0.01	0,909090909	0,969696970
0.1	0,878787879	0,960606061
1	0,030303030	0,109090909
10	0,030303030	0,109090909

Tabla 7: Precisión en función de la tolerancia

C

Este parámetro de regularización sirve como un grado de importancia que se le da a las clasificaciones incorrectas.

Se decide probar con más detalles variando C del algoritmo desde 0.0001 a 1000.0 y se conforma la siguiente tabla.

C	Number Accuracy	Leter Accuracy
0.00001	0,912121212	0,96969697
0.0001	0,912121212	0,96969697
0.001	0,912121212	0,96969697
0.01	0,912121212	0,96969697
0.1	0,912121212	0,96969697
1	0,912121212	0,96969697
10	0,912121212	0,96969697
100	0,912121212	0,96969697

Tabla 8: Precisión en función de c

Conclusiones preliminares

Con estos datos se concluye que los mejores parámetros son:

- Utilizar un Kernel lineal
- Con una cantidad máxima de iteraciones de 1
- Una tolerancia de 0.001
- Una C de 1

Parte 3

Con estos parámetros se decidió probar el algoritmo 400 veces con el conjunto de entrenamiento y de test para números y letras. Esto permite formar los siguientes gráficos, para así obtener la media de la precisión junto con su desviación estándar. Se adjunta un archivo llamado análisis_data.csv con los datos detallados.

Diagrama 9: Resultado final en 400 iteraciones

Además, se calcula:

- la media de las letras que resulta en 0.8976 con una desviación estándar de 0.0384
- la media de los números que resulta en 0.9622 con una desviación estándar de 0.0231

Se observa que el algoritmo para los números presenta una baja dispersión de la precisión posible. Indicando que el 96.22% es una buena aproximación del porcentaje de precisión. Además, presenta solo un dato apartado que podría deberse a una combinación de datos problemática.

En contraposición en las letras, el algoritmo presenta un menor caso de éxito y una dispersión del resultado mayor. Esto puede deberse a que se necesita un dataset mas amplio para obtener más casos de prueba.

Conclusiones

Con estos datos se concluye que el mejor algoritmo para poder resolver este problema con este dataset es el Support vector machine, usando los siguientes parámetros:

- Utilizar un Kernel lineal
- Con una cantidad máxima de iteraciones de 1
- Una tolerancia de 0.001
- Una C de 1

Esto permitiría obtener una precisión de 90% para letra y 96 % para números con una posibilidad de obtención de la patente completa sin errores de 66%.

Mejoras propuestas

Dataset

En primer lugar, se recomendaría la obtención y tratamiento de un dataset mayor. Esto permitiría aplicar otros enfoques como eliminar la necesidad de separa por caracteres y utilizar la patente completa. También esto permitiría el uso de otros algoritmos como las redes neuronales y un aumento de precisión de los algoritmos propuestos anteriormente.

Algoritmos

Se podría continuar aplicando algoritmos para la creación del xml que en este informe se hace manualmente. Esto permitiría que a partir de una foto de un auto se podría obtener la patente sin la necesidad de indicar en donde se encuentra la patente. Para esto seria recomendable usar redes neuronales para así obtener una mayor cantidad de relaciones más complejas. No solo evaluar la relación de los pixeles sino también de los colores, líneas, contornos, secciones de la imagen. Así se poder obtener más información para la toma de decisiones.

Luego se podría terminar aplicando este mismo enfoque a videos, separándolo en fotogramas. Permitiendo así la obtención de la patente a partir de una cámara de seguridad sin la intervención de una persona.

Bibliografía

[1] Russell, S. J. (2020). *Inteligencia Artificial Un Enfoque Moderno* (2.ª ed.). PRENTICE HALL/PEARSON.

[2] Scikit-learn. (2020). Support Vector Machines. Support Vector Machines. https://scikit-learn.org/stable/modules/svm.html

[3] Joaquín Amat Rodrigo j.amatrodrigo@gmail.com. (2020). Máquinas de Vector Soporte (Support Vector Machines, SVMs). Máquinas de Vector Soporte. https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines

[4] Patel, S. (2018, 10 noviembre). Chapter 1-2 : SVM (Support Vector Machine) — Theory - Machine Learning 101. Medium. https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72