Instituto Tecnológico y de Estudios Superiores de Monterrey.

Sistemas Inteligentes: Clasificación supervisada.

Por: Emmanuel Antonio Ramírez Herrera - A01703442

Introducción:

La agricultura es una actividad fundamental para la seguridad alimentaria, el desarrollo económico y la conservación ambiental de México. Aun así, la producción agrícola se enfrenta a diversos desafíos que amenazan su sostenibilidad y rentabilidad, como el cambio climático, las plagas, las enfermedades, la competencia de mercados externos y la falta de infraestructura y financiamiento, lo cual puede resultar en pérdidas significativas de los cultivos, afectando el ingreso de los productores y el abasto de alimentos para la población mexicana.

Según la Encuesta Nacional Agropecuaria 2019 del INEGI, en México se sembraron 21.5 millones de hectáreas de cultivos anuales y perennes, de las cuales se cosecharon 19.8 millones y se siniestraron 1.7 millones. Esto implica que el 8% de la superficie sembrada se perdió, lo que representa una disminución en la cantidad de cultivo generado y el valor de producción del mismo. Sin embargo, es bien sabido que este porcentaje puede variar según el tipo de cultivo, la región geográfica, las condiciones climáticas y las medidas de control de daños implementadas. En algunos casos, el índice de pérdida de cultivos puede superar el 20%, lo que se considera como un umbral crítico para la viabilidad económica y social de la actividad agrícola.

Ante este escenario, es necesario contar con herramientas que permitan predecir y prevenir las pérdidas de cultivos, así como optimizar el uso de los recursos disponibles. Una de estas herramientas es la inteligencia artificial (IA), ya que puede aplicarse a diversos problemas relacionados con la agricultura, como la detección de plagas, la estimación de rendimientos, la recomendación de insumos, la identificación de zonas potenciales y la clasificación de pérdidas.

Marco teórico:

La agricultura es una actividad fundamental para la seguridad alimentaria, el desarrollo económico y la conservación ambiental de México. Según el INEGI, en 2019 el sector primario aportó el 3.4% del Producto Interno Bruto (PIB) nacional y ocupó al 12.8% de la población económicamente activa (PEA). Además, México es uno de los principales productores y exportadores de productos agrícolas a nivel mundial, como maíz, frijol, trigo, jitomate, chile, cebolla, calabaza, soya, arroz y amaranto.

Como ya se mencionó previamente, en algunos casos, el índice de pérdida de cultivos que supera el 20%, se considera como un umbral crítico, por lo tanto, durante el desarrollo de este proyecto se tomará dicho índice como parte importante de las pruebas y el análisis de la información.

Dado que el aprendizaje supervisado consiste en entrenar algoritmos con datos etiquetados y el set de información solo contiene datos numéricos, sería importante realizar un tratamiento de la información previo a su uso para el entrenamiento. El objetivo es que el algoritmo pueda generalizar y predecir la salida para nuevos datos.

La clasificación consiste en asignar una categoría o clase a una entrada, por ejemplo, si una imagen contiene una plaga o no o en este caso, si un cultivo será pérdida o no. Esto se puede obtener mediante diferentes algoritmos, como árboles de decisión, regresión logística, redes neuronales artificiales, etc…

En este trabajo se propone desarrollar un modelo de IA que pueda predecir si una siembra será pérdida o no, según los diferentes parámetros proporcionados durante la consulta. El modelo se basará en técnicas de aprendizaje automático supervisado, que consisten en entrenar algoritmos con datos etiquetados para que puedan generalizar y clasificar nuevos datos. El objetivo es crear

un modelo que sea capaz de aprender patrones y relaciones a partir de los datos históricos y actuales de la producción agrícola en México (por lo menos hasta el año 2021), así como de variables externas que puedan influir en el resultado. El modelo se evaluará con diferentes métricas de desempeño y se comparará con otros modelos.

Metodología.

Es importante conocer la metodología utilizada para llevar a cabo este proyecto, puesto que de obtener resultados favorables, se debe documentar para su reproducción o en caso de obtener resultados no favorables, tener un histórico de los diversos intentos realizados para ayudar a subsanar esta necesidad latente.

El lenguaje utilizado para la realización de este proyecto es Python, por su facilidad de uso y por la cantidad de librerías de apoyo que ofrece, tales como pandas para el manejo de datos o numpy para cálculos matemáticos con vectores y matrices.

Obtención de datos.

Los datos han sido obtenidos por medio de la institución gubernamental del Servicio de Información Agroalimentaria y Pesquera, a través de su sitio web de datos abiertos. Específicamente se ha descargado la estadística de producción agrícola para el año 2021, donde este archivo se encuentra en formato CSV, lo cual permite un fácil manejo de la información contenida en el mismo.

Transformando la información para su uso.

La información obtenida contenía un total de 24 campos o atributos, de los cuales, cerca del 50% eran poco útiles para su uso en este problema en particular. Si bien es cierto que en la mayoría de los casos puede llegar a ser un inconveniente manejar identificadores numéricos para el entrenamiento de ciertas inteligencias artificiales, en este caso estos mismos pueden ayudarnos a facilitar y agilizar el entrenamiento de nuestros modelos, puesto que en su mayoría reciben datos categóricos en formato numérico, por lo que si se tienen datos con caracteres, estos suelen ser transformados por medio de algunos algoritmos como el LabelEncoder de scikit-learn. Pero al solo utilizar los identificadores, podemos ahorrarnos este proceso.

También es importante mencionar que como se espera predecir previo a que se tenga que realizar la siembra, tenemos que deshacernos de la información de la cantidad cosechada, el valor producido, o cualquier otro dato que en teoría tendríamos posterior al cultivo del producto.

Finalmente, nuestro campo de la cantidad siniestrada es el que nos va a ayudar con la categorización de cada una de las instancias del entrenamiento, pero este es un dato numérico, por lo que necesitamos convertirlo a un dato categórico. Para realizar esta conversión se hará uso del umbral previamente establecido, donde, si la cantidad de cultivo siniestrado es igual o mayor al 20% del cultivo sembrado, entonces se le asignará el label de que ha sido una pérdida.

La información final queda con tan solo 10 atributos, siendo estos: Año, Estado, Municipio, Ciclo productivo, Modalidad, Unidad, Cultivo, Sembrada, Siniestrada y Pérdida. (Recordemos que la mayoría de estos datos están representados con su identificador). Aquí un ejemplo de los datos de entrada:

```
(env) PS C:\Users\maner\Desktop\University\12o\Sistemas Inteligentes\Proyecto 3> python main.py
       Anio Idestado Idmunicipio Idciclo Idmodalidad Idunidadmedida Idcultivo Sembrada  Perdida
22166  2021       24          14       1           2         200201   7490000   231.25        0
7436   2021       13          44       2           2         200201   6840000      149        0
11613  2021       16         105       1           1         200201   5490000      200        0
21519  2021       24          35       1           1         200201   7330000       53        0
27069  2021       30         152       1           2         200201   8810000       14        1
...     ...      ...         ...     ...         ...            ...       ...      ...      ...
27115  2021       30         153       2           2         200201   7490000      253        0
3457   2021       10          26       3           2         200201   5060000      302        0
24316  2021       28          37       1           2         200201   8810000     2800        0
21730  2021       24          50       2           2         200201   7470000      108        1
12111  2021       16          22       3           1         200201   5170000       43        0

[714 rows x 9 columns]
```

Es necesario recalcar que lo que queremos predecir es el atributo de **Pérdida,** donde en este caso un 0 significa que no hay pérdida, pero un 1 significa que si**.**

Construyendo el perceptrón.

Para construir el perceptrón se requiere de conocer cómo funcionan los cálculos de manera interna en una IA, para esto lo más importante es conocer la función general que representa al perceptrón, la cual es $y = mx + b$, donde a partir de la implementación vectorizada de la misma (gracias a numpy) podemos aplicar las fórmulas requeridas para realizar los ajustes paramétricos necesarios (recordemos que y es lo que se desea predecir, x son las "variables de entrada" y "x" y "b" son los parámetros a ajustar).

Lo más importante al construir al perceptrón es implementar correctamente las funciones de forward y backward propagation, junto la de obtener el costo (que es donde se evalúa la entropía). Una vez implementado esto correctamente, podemos agregar las funciones extra para calcular precisión, graficar el entrenamiento y evaluar la efectividad del modelo (con una matriz de confusión, por ejemplo).

Construyendo una red neuronal (Tensorflow):

En este caso estamos haciendo uso de la librería Tensorflow de python, por lo que es bastante más sencillo el construir redes neuronales que como por ejemplo con lo que se hizo con el perceptrón.

Para resolver nuestro problema vamos a necesitar una red neuronal secuencial, porque queremos que los resultados de una capa se pasen directamente a la siguiente y que de esta forma la información fluya en una sola dirección. Pero dado a que necesitamos que las neuronas creen relaciones complejas entre sí, las conectaremos a todas las neuronas que existan en la capa anterior y siguiente, lo cual logramos con capas Dense.

Otra parte importante que debemos elegir al momento de construir este modelo es el tamaño de las capas, donde, entre más grandes sean, más características específicas y detalladas podrá extraer. En este caso se ha elegido algo moderado, obteniendo 64 -> 32 -> 1. Es importante mencionar que la última capa siempre debe tener como tamaño la cantidad de clases que vamos a predecir, siendo en este caso solo una "Pérdida".

También es importante que nuestra red neuronal utilice como función de coste "binary_crossentropy", puesto que deseamos clasificar una clase binaria y esta función de coste está optimizada para ello.

Finalmente, se crean las funciones extras para graficar y evaluar a nuestro modelo.

Construyendo un árbol de decisión (DecisionTreeClassifier).

En este caso, el DecisionTreeClassifier ya es un modelo preestablecido al que solo debemos pasar los parámetros necesarios para su entrenamiento, por lo tanto lo único importante es que la información proporcionada tenga un formato adecuado.

Lo último que queda con este solo es crear sus funciones para graficar y evaluar. En este caso lo importante es mencionar que la gráfica es en realidad la impresión del árbol de decisiones generado durante el entrenamiento.

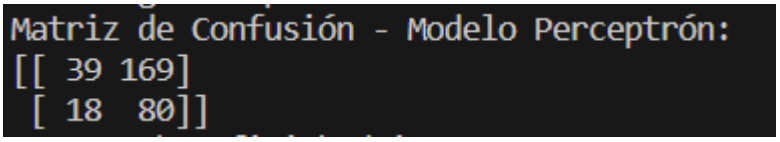
Resultados.

Se han ejecutado los 3 modelos con el mismo set de datos de entrenamiento y comprobando con el mismo set de datos de prueba, obteniendo resultados muy diferentes con cada uno de los modelos.

Perceptrón.

Tiempo de ejecución: 3.435 segundos

Matriz de confusión:



- Verdaderos Positivos (TP): 80
- Falsos Positivos (FP): 169
- Verdaderos Negativos (TN): 39
- Falsos Negativos (FN): 18

Con lo anterior calculamos las métricas:

- Accuracy = (TP + TN) / (TP + TN + FP + FN) = (80 + 39) / (80 + 39 + 169 + 18) = 0.2878
- Recall = TP / (TP + FN) = 80 / (80 + 18) = 0.8163
- Precision = TP / (TP + FP) = 80 / (80 + 169) = 0.3213
- F1-Score = 2 * (Precision * Recall) / (Precision + Recall) = 2 * (0.3213 * 0.8163) / (0.3213 + 0.8163) = 0.4595

Ventajas:

- Modelo sencillo y fácil de implementar.
- Eficiente computacionalmente para problemas linealmente separables.

Desventajas:

- No puede aprender funciones no lineales.
- Requiere un número de epochs para converger (terminar su entrenamiento).

Red neuronal (Tensorflow).

Tiempo de ejecución: 3.924 segundos

Matriz de confusión:

```
Matriz de Confusión - Modelo TensorFlow:
[[195  13]
 [ 72  26]]
```

- Verdaderos Positivos (TP): 26
- Falsos Positivos (FP): 13
- Verdaderos Negativos (TN): 195
- Falsos Negativos (FN): 72

Con lo anterior calculamos las métricas:

- Accuracy = (TP + TN) / (TP + TN + FP + FN) = (26 + 195) / (26 + 195 + 13 + 72) = 0.7222
- Recall = TP / (TP + FN) = 26 / (26 + 72) = 0.2653
- Precision = TP / (TP + FP) = 26 / (26 + 13) = 0.6667
- F1-Score = 2 * (Precision * Recall) / (Precision + Recall) = 2 * (0.6667 * 0.2653) / (0.6667 + 0.2653) = 0.3803

Ventajas:

- Capacidad para aprender funciones no lineales mediante capas ocultas.
- Flexibilidad para ajustar la arquitectura de la red (número de capas, unidades, etc.).

Desventajas:

- Requiere más datos y tiempo de entrenamiento para obtener buenos resultados.
- Posible sobreajuste si no se controlan adecuadamente los hiper parámetros

Árbol de decisión (DecisionTreeClassifier).

Tiempo de ejecución: 5.085 segundos

Matriz de confusión:

```
Matriz de Confusión - Modelo DecisionTreeClassifier:
[[174  34]
 [ 44  54]]
```

- Verdaderos Positivos (TP): 54
- Falsos Positivos (FP): 34
- Verdaderos Negativos (TN): 174
- Falsos Negativos (FN): 44

Con lo anterior calculamos las métricas:

- Accuracy = (TP + TN) / (TP + TN + FP + FN) = (54 + 174) / (54 + 174 + 34 + 44) = 0.7451
- Recall = TP / (TP + FN) = 54 / (54 + 44) = 0.5510
- Precision = TP / (TP + FP) = 54 / (54 + 34) = 0.6136
- F1-Score = 2 * (Precision * Recall) / (Precision + Recall) = 2 * (0.6136 * 0.5510) / (0.6136 + 0.5510) = 0.5800

Ventajas:

- Fácil interpretación del modelo en forma de árbol.
- Puede manejar datos faltantes y no lineales.

Desventajas:

- Sufre de sobreajuste si el árbol es demasiado profundo.
- Limitado en el número de divisiones y decisiones que puede hacer.

Conclusiones.

En este problema específico de clasificación de la columna "Perdida", el modelo DecisionTreeClassifier mostró el mejor rendimiento, con una mayor exactitud, precisión y un valor razonable de recall. Aunque el Perceptrón es un modelo simple y eficiente, no fue adecuado para este caso. Por otro lado, el modelo de red neuronal (Tensorflow) tiene una mayor capacidad para aprender debido a su escalabilidad, pero requiere más tiempo de entrenamiento y ajuste de hiper parámetros para obtener resultados óptimos.

En general, la elección del modelo depende de la complejidad del problema, la cantidad de datos disponibles y la interpretabilidad del modelo requerido. Si se busca un modelo fácilmente interpretable y el problema es relativamente simple, un árbol de decisión como el DecisionTreeClassifier podría ser una buena opción. Si el problema es más complejo, requiere de escalabilidad y se dispone de suficientes datos, un modelo de red neuronal como el implementado en Tensorflow podría proporcionar mejores resultados. Finalmente, el implementar la predicción mediante modelos personalizados como lo que se hizo con los perceptrones, lo dejaría meramente para fines académicos o de investigación.

Referencias.

- Cultivos y alimentos transgénicos en México: El debate, los actores y las fuerzas sociopolíticas. (2009). Revista Mexicana de Sociología, 71(1), 139-173. https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0187-57952009000100008
- El cambio climático aumentará el riesgo de propagación de plagas, que amenazan la seguridad alimentaria mundial - ONU. (2021, June 29). Noticias ONU. https://news.un.org/es/story/2021/06/1492762
- Mexico - Encuesta Nacional Agropecuaria 2019 - INEGI. (s.f.). INEGI. https://www.inegi.org.mx/rnm/index.php/catalog/607
- INEGI. (s.f.). Servicio de Información Agroalimentaria y Pesquera. Datos Abiertos. http://infosiap.siap.gob.mx/gobmx/datosAbiertos.php