How Severity the Accidents is?

CRISTIAN ALEXANDER CASTAÑO MONTOYA LUIS FELIPE CADAVID CHICA

Introducción a la Inteligencia Artificial / 2023-1

1. Introducción

Los accidentes de tránsito ocurren diariamente, vamos a analizar nuestro dataset, el dataset es un conjunto de datos de accidentes automovilísticos a nivel nacional que cubre 49 estados de los Estados Unidos. Los datos del accidente se recopilan desde febrero de 2016 hasta diciembre de 2021, utilizando múltiples API que proporcionan datos de incidentes (o eventos) de tráfico en tiempo real. Actualmente, hay alrededor de 4.2 millones de registros de accidentes en este conjunto de datos.

Para esto vamos a usar un dataset de Kaggle, que contiene 1.048.575 accidentes (filas) que será modificado según la necesidad del proyecto y 42 variables (columnas) que se puede observar a continuación:

Severity, Start_Time, End_Time, Start Start_Lng, End_Lat, End_ Distance(mi), Descri Number, Street Side, City Cc State Zipcode, Country, Time: Airport_Code, Weather_Timestamp, Temperature(F), Wind_Ch Humidity(%), Pressur Visibility(mi), Wind_Dire Wind_Speed(mph), Precipitation(in), Weather_Condition, Amenity,Bump, Cros Give_Way, Junction, No_Exit, Railway, Roundabout, Station,Stop, Traffic_Calming, Traffic_Signal, Turning_Loop ,Sunrise_Sunset, Civil_Twilight, Nautical_Twilight, Astronomical_Twilight

Nuestro problema de machine learning, es de clasificación, pues nuestra variable a predecir “Severity” está definida en 4 valores de 1 a 4 respectivamente, que definen que tan grave es el accidente. Teniendo en cuenta esto, nuestra métrica de machine learning va a ser el accuracy, el cual mide la proporción de predicciones correctas que realiza el modelo en comparación con el total de predicciones realizadas, expresada como porcentaje. Entre mayor sea el porcentaje de nuestro accuracy, mejor será el rendimiento del modelo. Por ejemplo, si el *severity* de un accidente es de 2 y nuestro modelo predice que es 3, y así sucesivamente con distintos datos, al final se sumará y obtendremos cuántos fueron las buenas predicciones y cuántas fueron las malas y obtendremos nuestro accuracy.

Debemos de tener un accuracy mínimo del 90%, solo así valdría la pena tener nuestro modelo funcionando. Que las predicciones que haga nuestro modelo después del entrenamiento, las buenas superen el 90%.

$$Accuracy = \frac{Correct\ prediction}{Total\ cases} * 100\%$$

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} * 100\%$$

Imagen 1. Formula del accuracy

2. Exploración descriptiva del dataset L

Este es un conjunto de datos de accidentes de tránsito de todo Estados unidos, que cubre 49 estados de este país. Los datos se recopilan continuamente desde febrero de 2016, utilizando varios proveedores de datos, incluidas varias API que brindan transmisión de datos de eventos de tráfico. Estas API transmiten eventos de tráfico capturados por una variedad de entidades, como los departamentos de transporte estatales y de EE. UU., agencias de aplicación de la ley, cámaras de tráfico y sensores de tráfico dentro de las redes de carreteras. Actualmente, hay alrededor de 1,5 millones de registros de accidentes en este conjunto de datos.

#	Atributo	Descripción
1	**ID**	Este es un identificador único del registro de accidentes.
2	**Severity**	Muestra la gravedad del accidente, un número entre 1 y 4, donde 1 indica el menor impacto en el tráfico (es decir, una breve demora como resultado del accidente) y 4 indica un impacto significativo en el tráfico (es decir, una gran demora).
3	**Start_Time**	Muestra la hora de inicio del accidente en la zona horaria local.
4	**End_Time**	Muestra la hora de finalización del accidente en la zona horaria local. La hora de finalización aquí se refiere a cuándo se descartó el impacto del accidente en el flujo de tráfico.
5	**Start_Lat**	Muestra la latitud en coordenadas GPS del punto de inicio.

6	**Start_Lng**	Muestra la longitud en coordenadas GPS del punto de inicio.
7	**End_Lat**	Muestra la latitud en coordenadas GPS del punto final.
8	**End_Lng**	Muestra la longitud en coordenadas GPS del punto final.
9	**Distance(mi)**	La longitud de la extensión de la carretera afectada por el accidente.
10	**Description**	Muestra la descripción del accidente en lenguaje natural.
11	**Number**	Muestra el número de calle en el campo de dirección.
12	**Street**	Muestra el nombre de la calle en el campo de dirección.
13	**Side**	Muestra el lado relativo de la calle (Derecha/Izquierda) en el campo de dirección.
14	**City**	Muestra la ciudad en el campo de dirección.
15	**County**	Muestra el condado en el campo de dirección.
16	**State**	Muestra el estado en el campo de dirección.
17	**Zipcode**	Muestra el código postal en el campo de dirección.
18	**Country**	Muestra el país en el campo de dirección.
19	**Timezone**	Muestra la zona horaria según la ubicación del accidente (este, centro, etc.).
20	**Airport_Code**	Indica una estación meteorológica en el aeropuerto que es la más cercana al lugar del accidente.
21	**Weather_Timestamp**	Muestra la marca de tiempo del registro de observación meteorológica (en hora local).
22	**Temperature(F)**	Muestra la temperatura (en Fahrenheit).
23	**Wind_Chill(F)**	Muestra la sensación térmica (en Fahrenheit).

24	**Humidity(%)**	Muestra la humedad (en porcentaje).
25	**Pressure(in)**	Muestra la presión del aire (en pulgadas).
26	**Visibility(mi)**	Muestra la visibilidad (en millas).
27	**Wind_Direction**	Muestra la dirección del viento.
28	**Wind_Speed(mph)**	Muestra la velocidad del viento (en millas por hora).
29	**Precipitation(in)**	Muestra la cantidad de precipitación en pulgadas, si hay alguna.
30	**Weather_Condition**	Muestra el estado del tiempo (lluvia, nieve, tormenta, niebla, etc.)
31	**Amenity**	Una anotación de PDI que indica la presencia de un servicio en un lugar cercano.
32	**Bump**	Una anotación de PDI que indica la presencia de badenes o jorobas en un lugar cercano.
33	**Crossing**	Una anotación de PDI que indica la presencia de un cruce en una ubicación cercana.
34	**Give_Way**	Una anotación de PDI que indica la presencia de “ceder el paso” en una ubicación cercana.
35	**Junction**	Una anotación de PDI que indica la presencia de un cruce en una ubicación cercana.
36	**No_Exit**	Una anotación de PDI que indica la presencia de “No salida” en una ubicación cercana.
37	**Railway**	Una anotación de PDI que indica la presencia de vías férreas en un lugar cercano.
38	**Roundabout**	Una anotación de PDI que indica la presencia de una rotonda en una ubicación cercana.
39	**Station**	Una anotación de PDI que indica la presencia de una estación en una ubicación cercana.

40	**Stop**	Una anotación de PDI que indica la presencia de una parada en una ubicación cercana.
41	**Traffic_Calming**	Una anotación de PDI que indica la presencia de “Calmante de tráfico” en una ubicación cercana.
42	**Traffic_Signal**	Una anotación de PDI que indica la presencia de “Señal de tráfico” en una ubicación cercana.
43	**Turning_Loop**	Una anotación de PDI que indica la presencia de “Bucle de giro” en una ubicación cercana.
44	**Sunrise_Sunset**	Muestra el período del día (es decir, día o noche) en función del amanecer/atardecer.
45	**Civil_Twilight**	Muestra el período del día (es decir, día o noche) según el crepúsculo civil.
46	**Nautical_Twilight**	Muestra el período del día (es decir, día o noche) basado en el crepúsculo náutico.
47	**Astronomical_Twiligh**	Muestra el período del día (es decir, día o noche) basado en el crepúsculo astronómico.

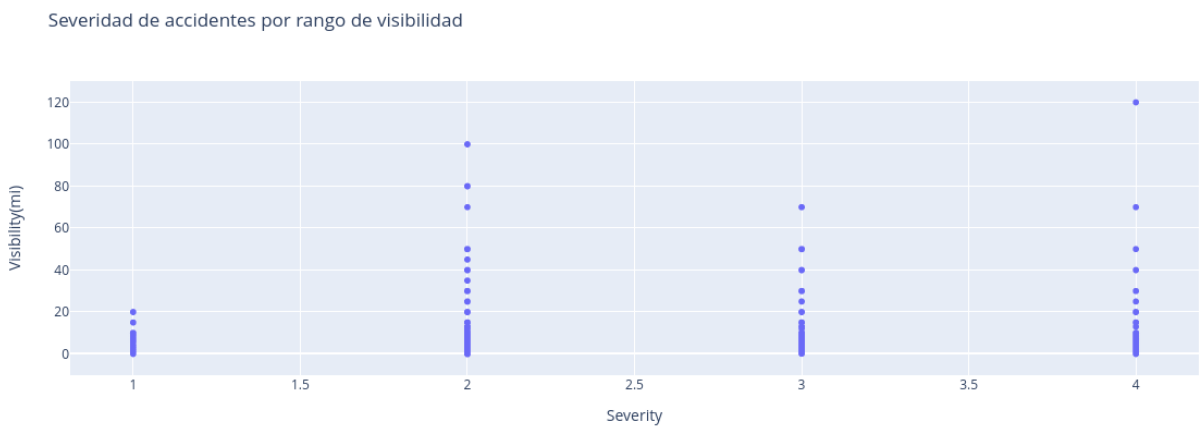
Tipo de datos encontrados

ID	object	Zipcode	object	Crossing	bool
Severity	int64	Country	object	Give_Way	bool
Start_Time	object	Timezone	object	Junction	bool
End_Time	object	Airport_Code	object	No_Exit	bool
Start_Lat	float64	Weather_Timestamp	object	Railway	bool
Start_Lng	float64	Temperature(F)	float64	Roundabout	bool
End_Lat	float64	Wind_Chill(F)	float64	Station	bool
End_Lng	float64	Humidity(%)	float64	Stop	bool
Distance(mi)	float64	Pressure(in)	float64	Traffic_Calming	bool
Description	object	Visibility(mi)	float64	Traffic_Signal	bool
Number	float64	Wind_Direction	object	Turning_Loop	bool
Street	object	Wind_Speed(mph)	float64	Sunrise_Sunset	object
Side	object	Precipitation(in)	float64	Civil_Twilight	object
City	object	Weather_Condition	object	Nautical_Twilight	object
County	object	Amenity	bool	Astronomical_Twilight	object
State	object	Bump	bool		

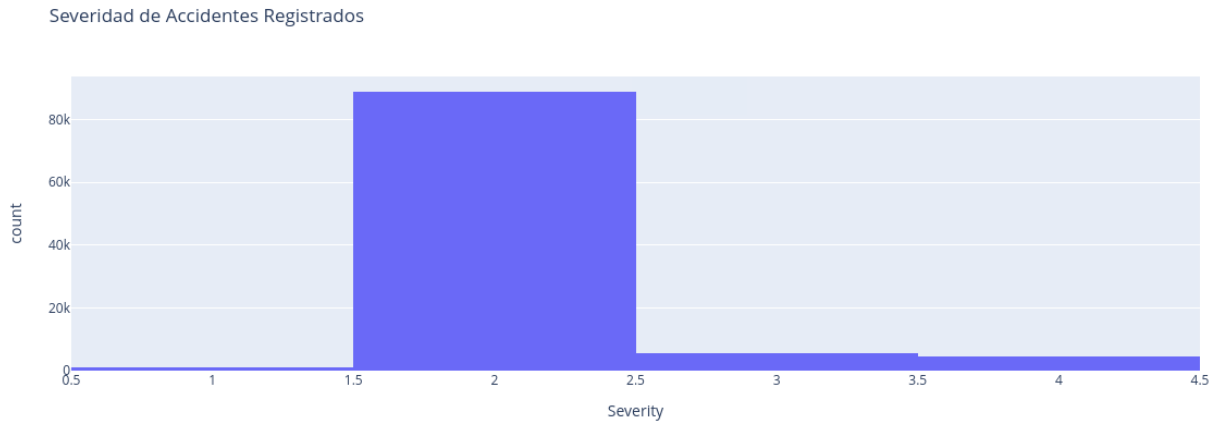
Valores nulos encontrados:

ID	0	Zipcode	44	Crossing	0
Severity	0	Country	0	Give_Way	0
Start_Time	0	Timezone	126	Junction	0
End_Time	0	Airport_Code	303	No_Exit	0
Start_Lat	0	Weather_Timestamp	1800	Railway	0
Start_Lng	0	Temperature(F)	2443	Roundabout	0
End_Lat	0	Wind_Chill(F)	16445	Station	0
End_Lng	0	Humidity(%)	2574	Stop	0
Distance(mi)	0	Pressure(in)	2101	Traffic_Calming	0
Description	0	Visibility(mi)	2518	Traffic_Signal	0
Number	61134	Wind_Direction	2603	Turning_Loop	0
Street	0	Wind_Speed(mph)	5579	Sunrise_Sunset	101
Side	0	Precipitation(in)	19158	Civil_Twilight	101
City	6	Weather_Condition	2528	Nautical_Twilight	101
County	0	Amenity	0	Astronomical_Twilight	101
State	0	Bump	0		

Severidad de Accidente por rango de visibilidad



Severidad de Accidente registrado:



3. Iteraciones de desarrollo. Para cada iteración, incluye los elementos que consideres de los siguientes:

- **Preprocesado de datos**

3.1 Preprocesado

El preprocesamiento de datos es una etapa fundamental antes de entrenar un modelo de aprendizaje automático. Se realiza para preparar y transformar los datos crudos de manera que sean adecuados y efectivos para el entrenamiento del modelo.

En nuestro caso, el preprocesado lo encontramos en el archivo "03_PRE_PROCESADO.ipynb" dentro del github. Lo primero que se hizo fue eliminar los datos duplicados, eso se realizó con la línea de código df.drop_duplicates(inplace**=True**). Para los valores faltaron decidimos eliminar todas las filas que no contenían algún valor, esto se hizo utilizando la línea de código df.dropna(inplace**=True**).

Con estos dos pasos, pasamos de tener 100.000 a 33306 rows, lo que nos hace ver que había muchos datos corrosivos. Con este nuevo dataset, hicimos una selección de características basada en la correlación. Esto es importante porque nos permite identificar y eliminar características redundantes o altamente correlacionadas en un conjunto de datos. Al tener características altamente correlacionadas, se puede introducir sesgo y ruido en el modelo, lo que puede afectar negativamente su rendimiento y capacidad de generalización. Además, la presencia de características redundantes aumenta la complejidad del modelo y puede llevar a un mayor tiempo de entrenamiento y mayores requisitos computacionales. Al seleccionar características basadas en su correlación, podemos reducir la dimensionalidad del conjunto de datos, mejorar la eficiencia y precisión del modelo, y obtener características más informativas y relevantes para la tarea de aprendizaje automático en cuestión.

Notamos un problema y era que nos dimos cuenta que las variables categóricas eran un problema al entrenar un modelo, pues nuestros modelos de clasificación funcionan mejor con variables numéricas. Para solucionar este problema, convertimos las variables categóricas en variables numéricas usando la codificación One-Hot.

Para finalizar el preprocesado, hicimos una normalización usando scaler = MinMaxScaler(), scaled_df = scaler.fit_transform(df). Con el fin de tener todos los datos estandarizados.

3.2 Modelos supervisados

Para hacer el entrenamiento usamos en total 5 modelos: Modelo Naive Bayes, modelo Random Forest, modelo KNN, modelo regresión logística y modelo Árboles de decisión.

3.2.1. Modelo Naive Bayes

El modelo Naive Bayes es un algoritmo de aprendizaje automático que se basa en el teorema de Bayes. Funciona asumiendo que todas las características son independientes entre sí, lo cual puede no ser cierto en la práctica, pero simplifica el cálculo. Durante el

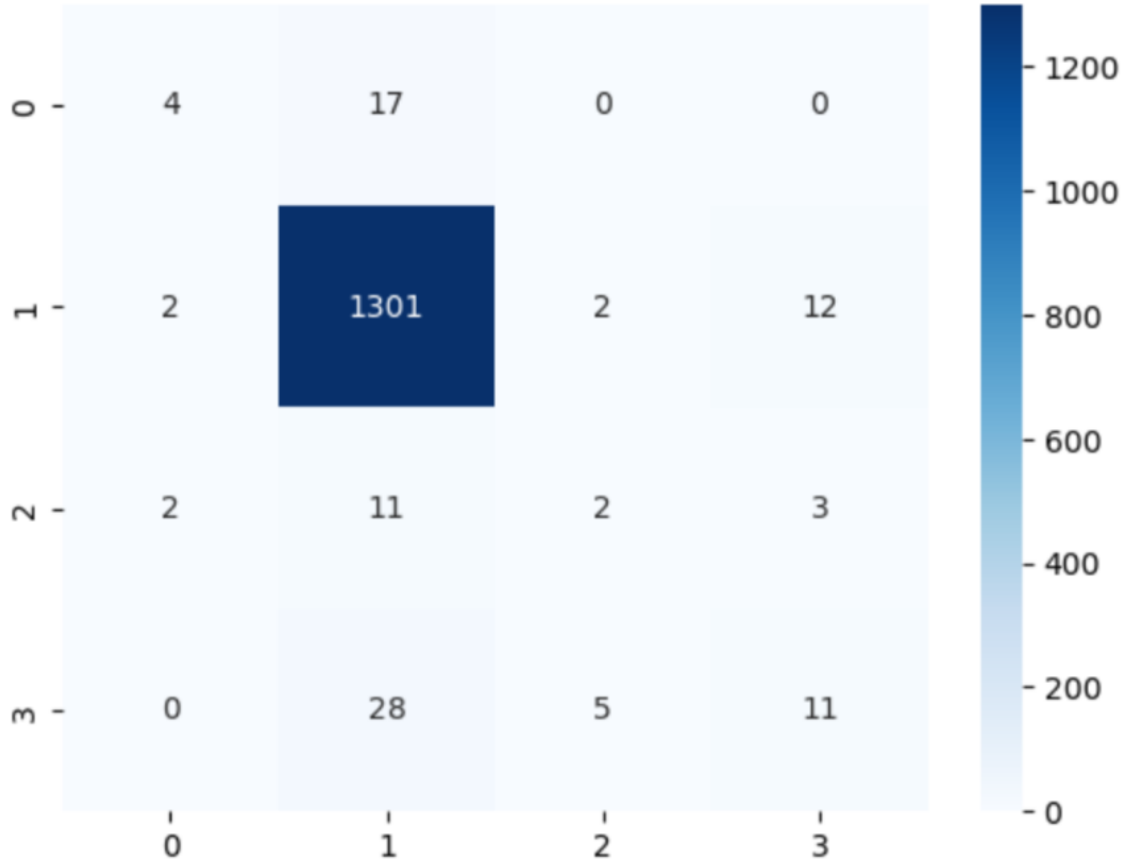
entrenamiento, el modelo calcula la probabilidad de cada clase y la probabilidad condicional de cada característica dada una clase. Luego, durante la predicción, utiliza estas probabilidades para calcular la probabilidad posterior de cada clase dado un nuevo ejemplo y elige la clase con la probabilidad más alta como la predicción.

Puedes encontrar el código completo en el archivo "04_MODELO_NAIVE_BAYES.ipynb". Para utilizarlo, se importaron las bibliotecas necesarias, se dividieron los datos en conjuntos de entrenamiento y prueba, se creó el modelo y se entrenó utilizando el método "fit". Luego, se obtuvieron métricas como la precisión, el recall, la matriz de confusión y la accuracy del modelo.

Los resultados de este modelo fueron:

- **Accuracy**: 0.94

- **Matriz de confusión:**



- **Precisión y Recall:**

```
Class 0: Precision: 0.50, Recall: 0.19
Class 1: Precision: 0.96, Recall: 0.99
Class 2: Precision: 0.22, Recall: 0.11
Class 3: Precision: 0.42, Recall: 0.25
```

De los resultados, podemos decir que el modelo se comportó bastante bien, los resultados fueron bastante buenos, con un accuracy del 94%. Sin embargo es necesario seguir revisando los otros modelos.

3.2.2. Modelo Random Forest

El modelo Random Forest es un algoritmo de aprendizaje automático que se basa en la técnica de ensemble learning, donde se combinan múltiples árboles de decisión para realizar predicciones. Funciona de la siguiente manera:

Durante el entrenamiento, se construye un conjunto de árboles de decisión utilizando diferentes muestras del conjunto de datos de entrenamiento y selecciones aleatorias de características. Cada árbol se entrena utilizando una combinación bootstrap del conjunto de datos y solo utiliza un subconjunto de características en cada división de nodo.

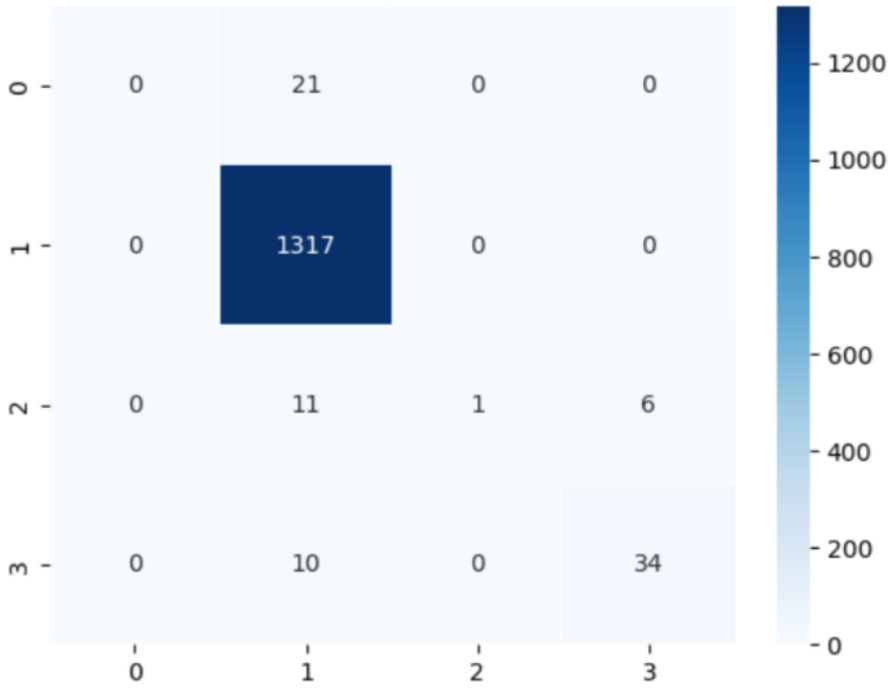
Durante la predicción, los resultados de los árboles individuales se combinan para obtener una predicción final. Esto se realiza mediante votación (en el caso de clasificación) o promediando (en el caso de regresión) las predicciones de los árboles.

Puedes encontrar el código completo en el archivo "05_MODELO_RANDOM_FOREST.ipynb". Para utilizarlo, se importaron las bibliotecas necesarias, se dividieron los datos en conjuntos de entrenamiento y prueba, se creó el modelo y se entrenó utilizando el método "fit". Luego, se obtuvieron métricas como la precisión, el recall, la matriz de confusión y la accuracy del modelo.

Los resultados de este modelo fueron:

- **Accuracy**:0.97

- **Matriz de confusión:**



- **Precisión y Recall:**

```
Class 0: Precision: 0.00, Recall: 0.00
Class 1: Precision: 0.97, Recall: 1.00
Class 2: Precision: 1.00, Recall: 0.06
Class 3: Precision: 0.85, Recall: 0.77
```

De este modelo, basado en los resultados y haciendo la comparación con el modelo anterior, vemos que es mucho más preciso. Aumentó el accuracy en un 3%, pasamos de tener un 94% a lograr un 97%.

3.2.3. Modelo KNN

El modelo K-Nearest Neighbors (KNN) funciona de la siguiente manera: Durante el entrenamiento, el modelo almacena todas las instancias de entrenamiento en una estructura de datos, como un árbol KD o una matriz de distancia. Cuando se recibe un nuevo ejemplo para predecir, el modelo encuentra los "K" vecinos más cercanos al ejemplo según alguna medida de distancia, como la distancia euclidiana.

Luego, el modelo utiliza la información de las etiquetas de los vecinos más cercanos para predecir la etiqueta del nuevo ejemplo. En el caso de clasificación, se realiza una votación entre los vecinos para determinar la etiqueta más común. En el caso de regresión, se realiza un promedio de los valores de los vecinos para obtener una predicción numérica.
El modelo KNN es simple y no requiere entrenamiento explícito, pero puede ser computacionalmente costoso al tener que calcular las distancias entre todos los ejemplos de

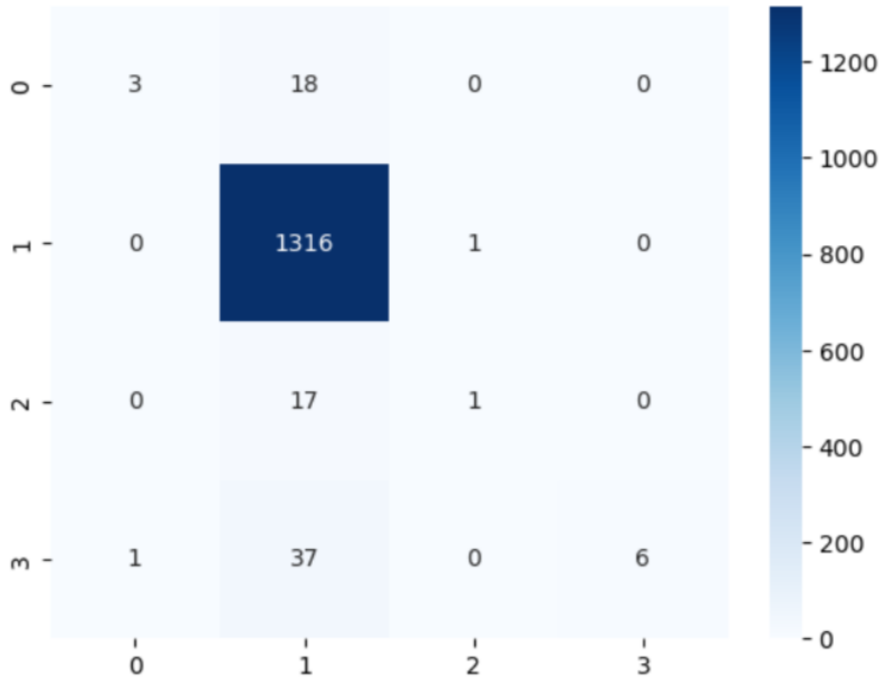
entrenamiento. Además, la elección del valor de "K" y la medida de distancia son importantes para obtener resultados precisos.

Puedes encontrar el código completo en el archivo "06_MODELO_K_NEAREST_NEIGHBORS.ipynb". Para utilizarlo, se importaron las bibliotecas necesarias, se dividieron los datos en conjuntos de entrenamiento y prueba, se creó el modelo y se entrenó utilizando el método "fit". Luego, se obtuvieron métricas como la precisión, el recall, la matriz de confusión y la accuracy del modelo.

Los resultados de este modelo fueron:

- **Accuracy**: 0.94

- **Matriz de confusión:**



- **Precisión y Recall:**

```
Class 0: Precision: 0.75, Recall: 0.14
Class 1: Precision: 0.95, Recall: 1.00
Class 2: Precision: 0.50, Recall: 0.06
Class 3: Precision: 1.00, Recall: 0.14
```

De este modelo, basado en los resultados y haciendo la comparación con el modelo anterior, vemos que es igual de eficiente que el modelo naive bayes y es menos eficiente que el modelo random forest.

3.2.4. Modelo regresión logística

El modelo de regresión logística funciona de la siguiente manera: Durante el entrenamiento, el modelo utiliza una función logística para modelar la relación entre las variables independientes y la probabilidad de pertenecer a una clase específica. Se ajustan los parámetros del modelo mediante el método de máxima verosimilitud, minimizando la función de costo, como la entropía cruzada.

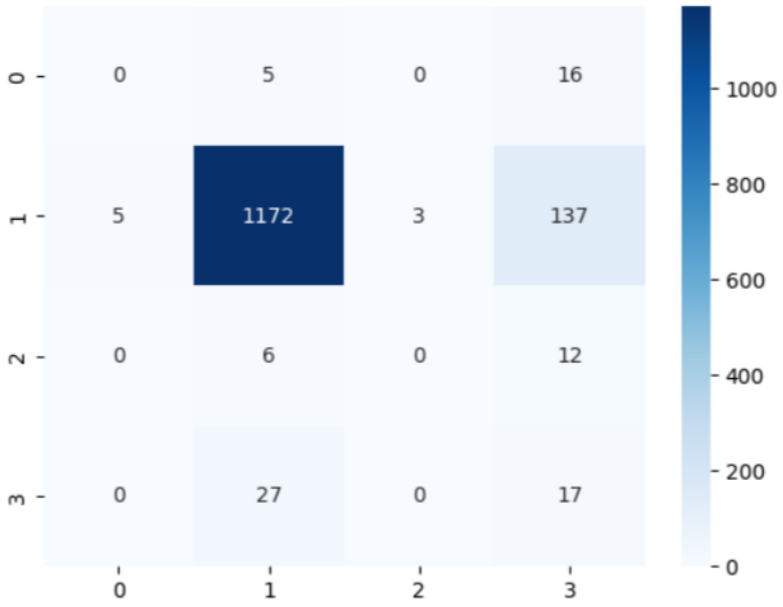
Una vez entrenado, el modelo puede realizar predicciones para nuevos ejemplos. Utiliza los parámetros aprendidos y aplica la función logística a las variables independientes del ejemplo para estimar la probabilidad de pertenecer a la clase objetivo. Luego, se aplica un umbral para determinar la clase final. Si la probabilidad estimada está por encima del umbral, se clasifica como la clase positiva; de lo contrario, se clasifica como la clase negativa.

Puedes encontrar el código completo en el archivo "07_MODELO_REGRESION_LOGISTICA.ipynb". Para utilizarlo, se importaron las bibliotecas necesarias, se dividieron los datos en conjuntos de entrenamiento y prueba, se creó el modelo y se entrenó utilizando el método "fit". Luego, se obtuvieron métricas como la precisión, el recall, la matriz de confusión y la accuracy del modelo.

Los resultados de este modelo fueron:

- **Accuracy**: 0.85

- **Matriz de confusión:**



- **Precisión y Recall:**

```
Class 0: Precision: 0.00, Recall: 0.00
Class 1: Precision: 0.97, Recall: 0.89
Class 2: Precision: 0.00, Recall: 0.00
Class 3: Precision: 0.09, Recall: 0.39
```

Este modelo tuvo un accuracy del 85%, sin embargo, se decidió aplicar un paso más, la extracción de características por PCA, con esto el accuracy aumentó al 94%. Sin embargo, aunque mejoró el accuracy sigue siendo mejor el modelo de random forest.

3.2.5. Modelo Árboles de decisión

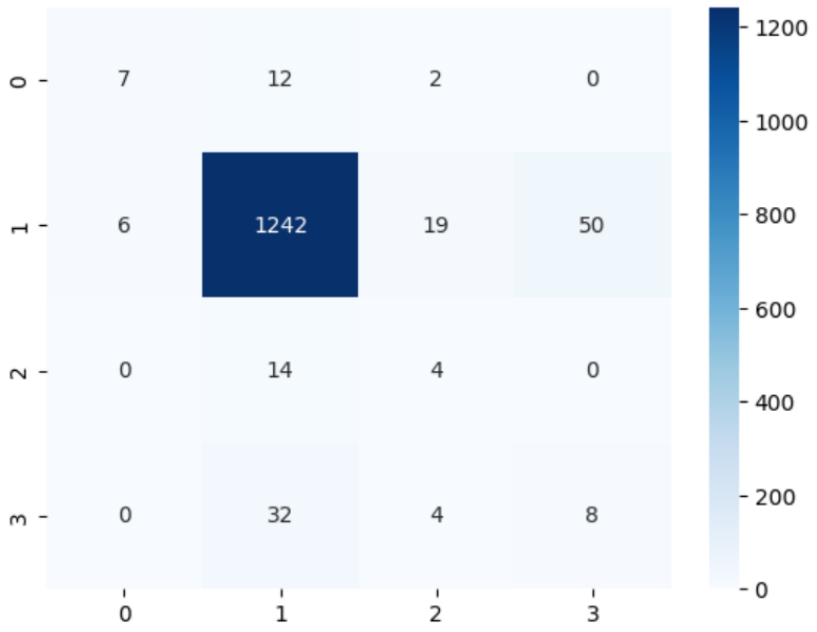
El modelo de Árboles de Decisión funciona de la siguiente manera: Durante el entrenamiento, el modelo construye una estructura de árbol donde cada nodo representa una pregunta sobre una característica del conjunto de datos. Se selecciona la pregunta que mejor separa los datos en función de alguna medida de impureza, como la ganancia de información o la impureza de Gini. A medida que se desciende por el árbol, se hacen sucesivas preguntas y se llega a hojas que representan las clases o valores de regresión finales. Durante la predicción, se sigue el camino desde la raíz hasta una hoja correspondiente al ejemplo dado, y se toma la clase o valor de regresión asociado.

Puedes encontrar el código completo en el archivo "08_MODELO_Árboles_de_decisión.ipynb". Para utilizarlo, se importaron las bibliotecas necesarias, se dividieron los datos en conjuntos de entrenamiento y prueba, se creó el modelo y se entrenó utilizando el método "fit". Luego, se obtuvieron métricas como la precisión, el recall, la matriz de confusión y la accuracy del modelo.

Los resultados de este modelo fueron:

- **Accuracy**: 0.90

- **Matriz de confusión:**



- **Precisión y Recall:**

```
Class 0: Precision: 0.54, Recall: 0.33
Class 1: Precision: 0.96, Recall: 0.94
Class 2: Precision: 0.14, Recall: 0.22
Class 3: Precision: 0.14, Recall: 0.18
```

En este modelo, sucedió algo muy particular. Inicialmente el accuracy dio un resultado de 0.03, demasiado bajo. Pero al aplicar la técnica de extracción de características por PCA aumentó a 0.90 el accuracy. Sin embargo sigue siendo bajo en comparación a los modelos anteriores.

3.3 Resultados, métricas y curvas de aprendizaje.

	Modelo Naive	**Modelo R forest**	**Modelo KNN**	**Modelo Reg Logística**	**Modelo árbo decisión**
Accuracy	0.94	0.97	0.94	0.94	0.90

Tabla 1. Accuracy de todos los modelos

Los resultados de los modelos indican la precisión o exactitud (accuracy) de cada modelo en la clasificación de los datos. El accuracy representa la proporción de ejemplos clasificados correctamente en relación con el total de ejemplos.

En base a los resultados proporcionados:

El modelo Random Forest obtuvo la mayor precisión con un 97% de accuracy, lo que indica que clasificó correctamente el 97% de los ejemplos en el conjunto de datos evaluado.

Tanto el Modelo Naive Bayes como el Modelo KNN obtuvieron una precisión de 94%. Esto significa que ambos modelos clasificaron correctamente el 94% de los ejemplos.

El Modelo de Árboles de Decisión mostró una precisión ligeramente más baja con un 90% de accuracy. Esto indica que clasificó correctamente el 90% de los ejemplos en el conjunto de datos evaluado.

En general, los modelos Random Forest y Naive Bayes, junto con el KNN, presentan una precisión similar y más alta que el Modelo de Árboles de Decisión.

4. Retos y consideraciones de despliegue

El despliegue efectivo y eficiente de este modelo implica varios consideraciones que es importante tener en cuenta, a continuación, mencionamos algunos de los aspectos clave:

Ambiente de producción: El modelo debe ser desplegado en un ambiente de producción acdecuado que pueda manejar las demandas de rendimientos y escalabilidad. Esto impolica considerar aspectos como la infraestructura necesaria, la capacidad de procesamiento, la disponibilidad de recursos y la integración con otros sistemas existentes.

Preparación de datos en el tiempo real: si el modelo requiere datos en tiempo real para hacer predicciones, es necsario implementar un flujo de datos eficiente que permita la captura, transformación y entrega de los datos de manera continua y en tiempo real al modelo desplegado

Monitorizaciño y mantenimiento: Una vez que el modelo está en producción, es importante monitorizar su rendimiento y asegurarse de que sigue siendo eficaz. Esto ultimo, implica realizar seguimiento de las métricas relevantes, detectar posibles problemas o degradación del rendimientoy realizar ajustes o actualizaciones según sea necesario. De igual manera, es importante mantener el modelo actualizado con nuevos datos.

Explicabilidad: Dentendiendo la funcionalidad para la cual sea adecuado el modelo, es importante tener en cuenta el nivel de explicabilidad que tiene las predicciones del modelo, y más si el escenario para el cual se acople el modelo, requiere de implementar desiciones criticas.

5. Conclusiones

En conclusión, al evaluar los resultados de los modelos basados en el accuracy, podemos observar que el Modelo Random Forest obtuvo la mayor precisión con un 97%, seguido por el Modelo Naive Bayes y el Modelo KNN, ambos con un 94% de accuracy. Por otro lado, el Modelo de Árboles de Decisión mostró una precisión ligeramente más baja, con un 90%.

Estos resultados resaltan la importancia de elegir el modelo adecuado para cada problema. El Modelo Random Forest demostró un rendimiento sobresaliente en este conjunto de datos, superando a los demás modelos. Sin embargo, es importante tener en cuenta que los resultados pueden variar según la naturaleza de los datos y las características del problema en cuestión.

Además, es fundamental resaltar la relevancia del pre-procesamiento de datos en el rendimiento de los modelos. Antes de entrenar cualquier modelo, es esencial realizar un preprocesamiento adecuado, que incluya la limpieza de datos, la normalización y el escalado, así como la codificación de variables categóricas. El preprocesamiento garantiza que los datos estén en la forma adecuada y pueda ser aprovechado al máximo por los modelos de aprendizaje automático.

En algunos casos, la extracción de características también puede ser beneficiosa para mejorar la precisión de los modelos. La técnica de extracción de características mediante PCA (Análisis de Componentes Principales) puede ayudar a reducir la dimensionalidad y capturar la variabilidad más relevante en los datos, lo que puede ser especialmente útil cuando hay un gran número de características.

En resumen, los resultados del accuracy muestran el rendimiento de los modelos en la clasificación de los datos. Sin embargo, es importante recordar que el preprocesamiento de datos, la estandarización y la aplicación de técnicas como la extracción de características por PCA pueden influir significativamente en el rendimiento y la precisión de los modelos de aprendizaje automático.

Referencias:

- **https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents?select=US_Accidents_Dec21_updated.csv**

- Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath. "A Countrywide Traffic Accident Dataset.", arXiv preprint arXiv:1906.05409 (2019).

- Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, and Rajiv Ramnath. "Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights." In proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2019.