Car Evaluation

Lourdes Najarro Cecilia Corzo Ana Hernández

[1]FISICC - SEMINARIO PROFESIONAL I [2]Universidad Galileo

1. Introducción

Este proyecto se basaba en redes neuronales y el entrenamiento de una para el análisis de un problema de Inteligencia Artificial, en esta oportunidad analizamos el data set de un problema llamado car evaluation para lo que se realizaron múltiples revisiones de los datos antes de comenzar con el entrenamiento de la red neuronal, a continuación, observaremos cada uno de los pasos que se realizaron para analizar el problema y para el entrenamiento de la red neuronal.

Así mismo vamos a mostrar que arquitectura nos proporcionó datos con mejores métricas y cuáles son nuestras mejoras a futuro en base a los resultados obtenidos y al proceso realizado.

2. Descripción del Dataset

Para dar un contexto el dataset proporcionado resuelve un problema de clasificación de automóviles según las características de cada auto, los clasifica en aceptable, buenos , muy buenos e inaceptables.

Teniendo el contexto podemos empezar a describir el dataset:

- ***Tiene un task de:*** *clasificación*
- *Contiene 1728 datos y ninguno de ellos tiene missing values*
- *Tiene un total de 6 features*
- *El archivo esta proporcionado en un formato CVS es decir separado por comas.*
- *Así mismo el dataset tiene todos los features de tipo categórico por lo que es necesario utilizar la técnica de preprocesamiento para convertir los datos a numéricos y no exista una dificultad más adelante al entrenarla.*
- *Tiene las variables de clasificación desequilibradas lo que nos indica que no existe suficiente información para la clasificación de los autos, aunque la mayoría de los datos si esta equilibrada.*
- *Existen dos features door y persons que son de tipo numérico y categórico por lo que los categóricos deben ser transformados a valores numéricos para una mayor eficiencia y entrenamiento.*
- ***Variables explicativas:*** *buying,maint,persons,lug_boot y safety.*
- ***Variables explicadas:*** *unacc,acc,Good y vgood.*

3. Metodología

Pasos para el entrenamiento de nuestro dataset:

1. ***P****reparación del entorno: Se instaló el paquete ucimlrepo que nos permitia acceder a los datasets del repositorio UCI Machine Learning Repository.* ***C****argar el dataset:descargamos el dataset para obtener los datos para su entrenamiento.*
2. ***E****xploración de los datos: Se observan las características de entrada y la variable objetivo. Se visualiza la distribución de clases, mostrando que el dataset está desbalanceado (por ejemplo, muchas muestras de unacc y pocas de vgood)*
3. ***P****reprocesamiento de la data: codificación One-Hot(Se codifican variables categóricas en forma numérica usando One-Hot Encoding. Cada categoría se convierte en una columna binaria.) Crear el pipeline(Esto transforma los datos en un formato apto para los modelos.)*
4. ***D****ivisión y balanceo:Se divide el dataset en entrenamiento y prueba (80Se aplica oversampling para balancear las clases del conjunto de entrenamiento. Esto evita que el modelo se sesgue hacia las clases más frecuentes.*
5. ***E****ntrenamiento de modelos, se utilizaron 3 modelos de entrenamiento. Modelo 1 Random Forest:Se entrena un clasificador basado en árboles aleatorios. Se calcula su accuracy y f1- score. Modelo 2Red Neuronal(MLP):Una red neuronal con tres capas ocultas ReLU, entrenamiento supervisado. Se evalúa como los demás Modelo 3 KNearest Neighbors:Modelo basado en los 5 vecinos más cercanos. Usa distancia entre puntos para clasificar.*
6. ***C****omparamos todos los modelos y escogimos el mejor según F1-Score.*
7. ***M****atrices de confusión:Muestra cuántos aciertos y errores cometió el modelo para cada clase.*
8. ***T****ransfer Learning:Se entrena primero en una muestra pequeña, y luego continúa el entrenamiento completo. Esto simula un tipo de aprendizaje progresivo.*

4. Resultados

A continuación veremos los resultados de los tres modelos de entrenamiento y cual es la que obtuvo mejores metricas. Tabla de las primeras tres arquitecturas:

Arquitectura	*Accuracy*	*F-Score*
Random Forest(100 árboles)	*0.9653465*	*0.9672131*
MLP(3 capas ReLU)	***0.9913043***	***0.9915254***
KNN(5 vecinos)	*0.7282608*	*0.7644230*

La tabla anterior nos mostraba que las metricas mejores son las que se realizaron con la arquitectura MLP(3 capas de ReLU) Tabla de las arquitc-

turas incluyendo la de transfer Learning

Arquitectura	Accuracy 2	F-Score
Random Forest(100 árboles)	0.9653465	0.9672131
MLP(3 capas ReLU)	**0.9913043**	**0.9915254**
KNN(5 vecinos)	0.7282608	0.7644230
Transfer Learning(MLP - 3 capas ReLU)	0.9913294	0.9914936

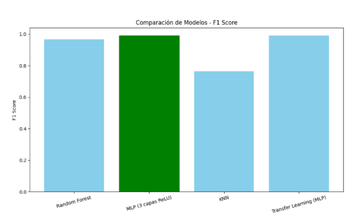

Figure 1. Comparacion de modelos

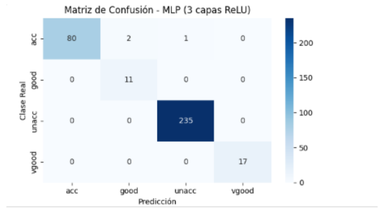

Figure 2. Matriz de confusión del mejor modelo

5. Conclusiones

Para el entrenamiento de este dataset se preprocesaron los datos aplicando One-Hot Encoding y se manejó el desbalance con oversampling. Se compararon tres modelos: Random Forest, MLP y KNN, siendo la red neuronal multicapa (MLP) la que obtuvo el mejor rendimiento en términos de F1-score y accuracy. También se exploró el uso de transfer learning para mejorar el entrenamiento del modelo MLP.
Finalmente, las matrices de confusión y las gráficas permitieron visualizar y confirmar la superioridad del modelo MLP, destacando su capacidad para clasificar correctamente las distintas categorías.

6. Mejoras a futuro

Para este dataset las primeras mejoras a futuro seria tener un amplio set de datos para realizar mejor las clasificaciones de los autos, ejemplos carcteristicas de motores para poder clasifcar mejor los autos. Tambien colocar los datos mas balanceados para disminuir el preprocesamiento de la data y obtener mejores metricas. Y por ultimo el poder realizar mas modelos de entrenamiento y compararlos para encontrar el mas certero.