INSTITUTO SUPERIOR POLITÉCNICO DE CÓRDOBA

TECNICATURA SUPERIOR EN CIENCIA DE DATOS E INTELIGENCIA ARTIFICIAL

Módulo de Práctica Profesionalizante

Informe Técnico

Entrega 4 - 28/10/2024

Docentes:

- Charletti, Carlos

Integrantes:

- López, Erick
- Nüesch, Christian
- Zurita Rojo, Débora

Índice

Índice.....1
Resumen.....3
Introducción.....3
Metodología.....3
Desarrollo.....3
Resultados y análisis.....6
Conclusiones.....7
Referencias.....8

Resumen

El presente informe técnico sintetiza el trabajo realizado en el curso de **Práctica Profesionalizante I**, documentando el proceso de carga y revisión de datos sobre ventas de vehículos automotores, así como la evaluación de tres modelos de aprendizaje automático y el análisis de los resultados obtenidos.

Introducción

Este trabajo se desarrolla en el marco de un espacio curricular que simula un entorno empresarial dedicado al análisis de datos, la empresa **DataVista Analytics**, con el objetivo de abordar la carencia de análisis de información en tiempo real para la toma de decisiones en la industria automotriz.

La estructura del trabajo incluyó la carga de librerías y configuración del entorno, una preparación básica del conjunto de datos, la evaluación de tres modelos de aprendizaje automático, un análisis comparativo de sus resultados y un ajuste detallado de los parámetros de uno de los modelos.

Metodología

La metodología de trabajo utilizada fue un entorno tipo Notebooks de **Jupyter/Colab**, ya que se basa en un enfoque interactivo que permite combinar celdas de código y texto en un solo documento. Por tanto, se pueden escribir y ejecutar bloques de código de manera independiente, lo que facilita la experimentación y la depuración, mientras que las celdas de texto en formato Markdown permiten documentar el proceso y explicar los resultados. Esta estructura favoreció la ejecución y la colaboración grupal en tiempo real, mostrando resultados inmediatos y permitiendo ajustes dinámicos en los métodos utilizados.

Desarrollo

En esta sección, se presenta un resumen de las diversas partes del código implementadas y ejecutadas en etapas sucesivas. Para obtener información más detallada, se sugiere consultar el trabajo completo, donde se proporciona una descripción más exhaustiva y comentarios sobre el código.

1. Configuramos un entorno de trabajo en Google Colab para el análisis de datos utilizando la biblioteca Pandas. Primero, se habilita el formato de visualización de tablas para los DataFrames mediante `data_table.enable_dataframe_formatter()`. Luego, se importa la biblioteca Pandas y se establece una opción para mostrar números flotantes con dos decimales, utilizando `pd.set_option('display.float_format', lambda x: '%.2f' % x)`. Esto permite que todos los números flotantes en las salidas de Pandas se presenten con un formato más legible y estandarizado, mejorando la claridad en la visualización de datos.
2. Cargamos el conjunto de datos en un DataFrame de Pandas llamado `data`. Se proporcionan dos opciones para la carga del dataset: la primera permite cargarlo desde el entorno local mediante un archivo CSV, que debe ser activada al comentar la línea correspondiente; la segunda opción, que está activa, lee el archivo directamente desde una URL en GitHub. Finalmente, incluimos una línea para verificar el contenido del DataFrame 'data', mostrando así los datos cargados.
3. Establecemos una tasa de cambio de 1 USD a 380 ARS y definimos una función llamada `convertir_a_dolar` que convierte precios de pesos a dólares, dependiendo del valor de la columna `Moneda` en cada fila del DataFrame. Si la moneda es `pesos`, el precio se divide por la tasa de cambio; de lo contrario, se mantiene sin cambios. A continuación, se aplica esta función a cada fila del DataFrame `data`, redondeando los resultados a dos decimales. Posteriormente, se eliminan las columnas `Moneda` y `Año_zscore` del DataFrame, y finalmente, verificamos el contenido del DataFrame transformado.
4. Utilizamos la función `train_test_split` de la biblioteca `sklearn.model_selection` para dividir el conjunto de datos en conjuntos de entrenamiento y prueba. Primero, se seleccionan las características (X) excluyendo la columna `Precio`, que se define como la variable objetivo (y). Luego, se realiza la división del dataset, asignando el 80% de los datos a los conjuntos de entrenamiento (`X_train` y `y_train`) y el 20% restante a los conjuntos de prueba (`X_test` y `y_test`), utilizando un parámetro `random_state` para asegurar la reproducibilidad del proceso. Esto permite evaluar el rendimiento del modelo de manera efectiva al entrenarlo con un conjunto de datos y probarlo con otro.
5. **Regresión lineal múltiple:** implementamos un modelo de regresión lineal utilizando la biblioteca `scikit-learn`, comenzando por la definición de las columnas predictoras y la variable objetivo a partir del conjunto de datos previamente cargado. Se identifican las características categóricas y numéricas, y se aplica un proceso de codificación para las variables categóricas mediante `LabelEncoder`, que incluye una función personalizada

para manejar valores desconocidos durante la transformación. A continuación, se crea un preprocesador utilizando `ColumnTransformer` para mantener las características numéricas sin cambios. Posteriormente, se establece un `Pipeline` que integra el preprocesador y el modelo de regresión lineal. El modelo se entrena con los datos de entrenamiento y se realizan predicciones sobre el conjunto de prueba. Finalmente, se evalúa el rendimiento del modelo utilizando métricas como el error cuadrático medio (MSE), el error absoluto medio (MAE) y el coeficiente de determinación (R^2), imprimiendo los resultados obtenidos.

6. En otra implementación de **regresión lineal múltiple** aplicamos One-Hot Encoding a las columnas categóricas del conjunto de datos, identificando previamente las características categóricas y numéricas. Se definen las columnas categóricas así como las columnas numéricas. A continuación, se crea un preprocesador utilizando `ColumnTransformer`, que aplica el codificador One-Hot a las variables categóricas mientras mantiene las columnas numéricas sin cambios mediante el parámetro `remainder='passthrough'`. Esto nos permitiò transformar adecuadamente los datos para su posterior uso en el modelo de aprendizaje automático, asegurando que las variables categóricas sean representadas de manera adecuada.
7. **Gradient Boosting:** implementamos un modelo de regresión utilizando el algoritmo de Gradient Boosting para predecir precios a partir del conjunto de datos. Primero, definimos las características (X) excluyendo la columna `Precio`, que se establece como la variable objetivo (y). Se identifican las columnas categóricas y numéricas, y se crea un preprocesador mediante `ColumnTransformer` que aplica One-Hot Encoding a las variables categóricas, ignorando categorías no vistas, mientras mantiene las columnas numéricas sin cambios. A continuación, se construye un `Pipeline` que integra el preprocesador y el modelo de Gradient Boosting, configurado con 100 estimadores, una tasa de aprendizaje de 0.1 y una profundidad máxima de 3. El conjunto de datos se divide en entrenamiento y prueba (80% y 20%, respectivamente), y el modelo se entrena con los datos de entrenamiento. Posteriormente, se realizan predicciones sobre el conjunto de prueba y se evalúa el rendimiento del modelo utilizando métricas como el error cuadrático medio (MSE), el error absoluto medio (MAE) y el coeficiente de determinación (R^2), imprimiendo los resultados obtenidos.
8. **Random Forest:** el último modelo que implementamos es un modelo de regresión utilizando el algoritmo de Random Forest para predecir precios a partir del conjunto de datos. Definimos las características (X) excluyendo la columna `Precio`, que se establece como la variable objetivo (y). Se identificaron las columnas categóricas y numéricas, y se

creó un preprocesador mediante `ColumnTransformer`, que aplica One-Hot Encoding a las variables categóricas, ignorando categorías no vistas y manteniendo las columnas numéricas sin cambios. A continuación, se construyó un `Pipeline` que integra el preprocesador y el modelo de Random Forest, configurado con 100 estimadores y un estado aleatorio para asegurar la reproducibilidad. El conjunto de datos se dividió en entrenamiento y prueba (80% y 20%, respectivamente), y el modelo se entrenó con los datos de entrenamiento. Posteriormente, se realizaron predicciones sobre el conjunto de prueba y se evaluó el rendimiento del modelo utilizando métricas como el error cuadrático medio (MSE), el error absoluto medio (MAE) y el coeficiente de determinación (R^2), imprimiendo los resultados obtenidos.

Resultados y análisis

Aquí se presenta el rendimiento de cada modelo comparando las métricas obtenidas:

	MSE	MAE	R^2
Regresión lineal múltiple - Primera prueba	1424760535.80	10918.03	0.36
Regresión lineal múltiple - Tercera prueba	786232977.76	8405.88	0.65
Gradient Boosting	657591633.06	6342.08	0.71
Random Forest	956274644.38	7212.61	0.57

Por último, interpretamos los resultados del modelo en el contexto del negocio automotriz, analizando su utilidad para la toma de decisiones y sus limitaciones. Las predicciones de precios de vehículos usados son cruciales para concesionarios y plataformas de venta, ya que permiten ajustar precios, formular estrategias de compra y establecer precios competitivos.

1. **MSE (Error Cuadrático Medio):** El modelo de Gradient Boosting presenta el menor MSE (657 millones), lo que indica que sus predicciones son más precisas, ayudando a evitar errores en la valoración de automóviles. La regresión lineal muestra mejoras, pero sigue siendo menos precisa.

2. **MAE (Error Absoluto Medio):** Con un MAE de 6342,08, Gradient Boosting también es el más preciso. Este error puede ser aceptable o no, dependiendo del precio promedio de los vehículos; es más significativo en autos de menor valor. Comparado con la regresión lineal (8405,88) y Random Forest (7212,61), el Gradient Boosting se destaca.
3. **R^2 (Coeficiente de determinación):** Gradient Boosting explica el 71% de la variabilidad en los precios, lo que indica un buen ajuste del modelo. La regresión lineal tiene un R^2 de 0.65 y Random Forest un 0.57, sugiriendo que este último es menos confiable para decisiones críticas.

En resumen, el modelo de Gradient Boosting se muestra como la mejor opción para predecir precios en el negocio automotriz, aunque cada modelo tiene sus propias limitaciones y aplicaciones.

Conclusiones

Gradient Boosting se destacó como el mejor modelo, presentando el menor MSE, MAE y el mayor R^2. En el ámbito empresarial, su uso podría resultar en predicciones de precios más precisas para automóviles, optimizando ganancias, ajustando estrategias de precios y minimizando errores significativos.

Impacto en el negocio: La precisión en la predicción de precios ayuda a evitar tanto la sobrevaloración, que puede impedir ventas, como la subvaloración, que puede resultar en pérdidas. Un menor error promedio con Gradient Boosting permite a la empresa establecer precios más competitivos, lo que podría aumentar las ventas y reducir riesgos financieros.

Posibles mejoras o consideraciones: Si los errores de predicción son aún relevantes, se podrían ajustar los hiperparámetros del modelo o explorar técnicas como el stacking de modelos y el uso de más datos disponibles. Además, la calidad de los datos es crucial; contar con información adicional sobre ventas y demanda podría mejorar aún más la precisión del modelo.

Referencias

Referencias bibliográficas:

Géron, A. (2020). Aprende Machine Learning con Scikit-Learn, Keras y TensorFlow: conceptos, herramientas y técnicas para construir sistemas inteligentes. Anaya Multimedia.

Ciencia de Datos. (2024). Material formativo sobre estadística, algoritmos de machine learning, ciencia de datos y programación en R y Python. https://cienciadedatos.net/

Apuntes de la materia

Bibliotecas externas utilizadas (por orden de aparición):

Google. (s.f.). Google Colaboratory (s.f.) [Software]. Google. https://colab.research.google.com

McKinney, W. (2010). pandas: a foundational Python library for data analysis (2024) [Software]. GitHub. https://pandas.pydata.org/

Cournapeau, D. (2011). scikit-learn: Machine Learning in Python (Versión 0.24.2) [Software]. Journal of Machine Learning Research. https://scikit-learn.org/

Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://numpy.org/

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90-95. https://matplotlib.org/

Waskom, M. L. (2021). seaborn: statistical data visualization (Versión 0.11.1) [Software]. GitHub. https://seaborn.pydata.org/