UNIVERSIDAD DEL VALLE DE GUATEMALA

Machine Learning
Sección 10

Lab 1

Juan Ángel Carrera 20593
Juan Carlos Baján 20109
José Mariano Reyes 20074
Esteban Aldana Guerra 20591
Luis Pedro Gonzalez Aldana 20008

GUATEMALA, 11 de Agosto de 2024

Ejercicio 1

Configuramos Visual Studio Code (VS Code) para un correcto orden del proyecto.

Ejercicio 2 (EDA)

El Análisis Exploratorio de Datos (EDA) realizado en el proyecto abarcó varios aspectos clave para entender y preparar los datos. A continuación, se presenta un resumen general de los pasos y análisis efectuados:

Configuración y Carga de Datos:

- Se importaron las bibliotecas necesarias, como pandas, numpy, matplotlib, seaborn, entre otras.
- Se cargaron los datos de entrenamiento y prueba desde archivos CSV.

Descripción General de los Datos:

- Se realizó una descripción general de las variables, incluyendo un resumen estadístico para todas las columnas, tanto numéricas como categóricas.
- Se analizó la presencia de valores nulos en el conjunto de datos, utilizando tanto un análisis numérico como visual mediante un mapa de calor.

Visualización de Distribuciones:

- Se generaron histogramas para visualizar la distribución de las variables numéricas. Esto ayudó a identificar características como la asimetría y la presencia de posibles valores atípicos.
- Se realizaron gráficos de conteo para las variables categóricas, proporcionando una visión de la frecuencia de cada categoría.

Análisis de Correlación:

- Se calculó y visualizó la matriz de correlación entre las variables numéricas, utilizando un mapa de calor para identificar relaciones fuertes o débiles entre ellas.
- Este análisis preliminar permitió identificar la calidad de los datos, entender sus características básicas y establecer una base para futuras etapas del análisis, como la limpieza de datos y la ingeniería de características.

Entrenamiento del Modelo de Red Neuronal

Descripción General

En este trabajo, desarrollamos y entrenamos un modelo de red neuronal para predecir los precios de casas utilizando un conjunto de datos con diversas características de las propiedades. El objetivo principal es construir un modelo inicial y evaluar su rendimiento.

Pasos Realizados

1. **Instalación de Dependencias**:
    - Se instalaron las bibliotecas necesarias, incluyendo pandas, numpy, scikit-learn, tensorflow y matplotlib.
2. **Carga y Preprocesamiento de Datos**:
    - **Carga de Datos**: Se cargaron los datos desde el archivo `train.csv`.
    - **Imputación de Valores Nulos**: Se imputaron los valores nulos con la media de las columnas numéricas para asegurar que no haya valores faltantes.
    - **Selección de Características y Variable Objetivo**: Se seleccionaron las características relevantes (`OverallQual`, `GrLivArea`, `GarageCars`, `GarageArea`, `TotalBsmtSF`) y la variable objetivo (`SalePrice`).
    - **Estandarización**: Las características seleccionadas fueron estandarizadas para mejorar el rendimiento del modelo.
3. **División de Datos**:
    - Los datos fueron divididos en conjuntos de entrenamiento y prueba utilizando una proporción del 80% para entrenamiento y 20% para prueba.
4. **Definición y Compilación del Modelo**:
    - **Arquitectura del Modelo**: Se definió un modelo de red neuronal con dos capas ocultas (64 y 32 neuronas, respectivamente) y una capa de salida.
    - **Compilación**: El modelo fue compilado utilizando el optimizador `adam` y la función de pérdida `mse` (mean squared error).
5. **Entrenamiento del Modelo**:
    - El modelo fue entrenado durante 100 épocas, utilizando el conjunto de datos de entrenamiento y validación.
6. **Evaluación del Modelo**:
    - **Predicciones**: Se realizaron predicciones sobre el conjunto de datos de prueba.
    - **RMSE**: Se calculó el RMSE para evaluar el rendimiento del modelo.
    - **Gráficas**:
        - **Pérdida durante el Entrenamiento**: Se visualizó la pérdida en los datos de entrenamiento y validación a lo largo de las épocas.
        - **Predicciones vs Valores Reales**: Se graficaron las predicciones frente a los valores reales para analizar el desempeño del modelo.
7. **Guardado de Resultados**:
    - **Modelo**: El modelo entrenado fue guardado en la carpeta `models/` como `nn_model.h5`.
    - **Resultados y Gráficas**: Los resultados del entrenamiento y las gráficas fueron guardados en la carpeta `reports/`.

Ajuste y Mejora del Modelo de Red Neuronal

Descripción General

En este trabajo, ajustamos el modelo de red neuronal previamente entrenado para mejorar su rendimiento. Implementamos técnicas de regularización y utilizamos un callback para detener el entrenamiento si no se observan mejoras, con el fin de evitar el sobreajuste.

Pasos Realizados

1. **Instalación de Dependencias**:
    - Se aseguraron las instalaciones de las bibliotecas necesarias, tales como pandas, numpy, scikit-learn, tensorflow y matplotlib.
2. **Carga y Preprocesamiento de Datos**:
    - **Carga de Datos**: Se cargaron nuevamente los datos desde `train.csv`.
    - **Imputación y Estandarización**: Similar al entrenamiento inicial, se imputaron valores nulos y se estandarizaron las características seleccionadas.
3. **División de Datos**:
    - Los datos fueron divididos en conjuntos de entrenamiento y prueba con la misma proporción de 80% y 20%, respectivamente.
4. **Definición del Modelo con Regularización**:
    - **Arquitectura Mejorada**: Se incrementó la complejidad del modelo con más neuronas (128 y 64 neuronas en las capas ocultas) y se añadió regularización mediante capas de Dropout para reducir el sobreajuste.
    - **Compilación**: Utilizando el optimizador `adam` y la función de pérdida `mse`.
5. **Implementación de Early Stopping**:
    - Se utilizó un callback de EarlyStopping para detener el entrenamiento si la pérdida de validación no mejora durante 10 épocas consecutivas, restaurando los pesos del mejor modelo obtenido durante el entrenamiento.
6. **Entrenamiento del Modelo**:
    - El modelo mejorado fue entrenado hasta un máximo de 200 épocas, con la posibilidad de detenerse antes si se activaba el early stopping.
7. **Evaluación del Modelo**:
    - **Predicciones y RMSE**: Se evaluó el modelo calculando el RMSE en el conjunto de prueba.
    - **Gráficas**:
        - **Pérdida durante el Entrenamiento (con Ajuste)**: Se visualizó la pérdida durante el entrenamiento y validación.
        - **Predicciones vs Valores Reales (con Ajuste)**: Se graficaron las predicciones ajustadas frente a los valores reales.
8. **Guardado de Resultados**:
    - **Modelo Ajustado**: El modelo ajustado fue guardado en la carpeta `models/` como `nn_model_tuned.h5`.
    - **Resultados y Gráficas**: Los resultados del ajuste y las gráficas fueron guardados en la carpeta `reports/`.

Conclusión

Los ajustes realizados, incluyendo la regularización y el uso de early stopping, buscaron mejorar la capacidad de generalización del modelo y prevenir el sobreajuste, resultando en un rendimiento más robusto y confiable. Al implementar estas técnicas, el modelo mostró

una mejor alineación con los datos de validación, lo que sugiere que está mejor preparado para predecir sobre datos nuevos. Además, el uso de regularización y la estrategia de early stopping redujeron el tiempo de entrenamiento, asegurando que el modelo no se entrenara en exceso.