**Clara Victoria García Chávez**

**Informe: Predicción de precios de aguacate con machine learning y deep learning**

**Subclase elegida**

Se seleccionó el aguacate como producto de la canasta básica para este análisis. Esta elección se justifica por la alta volatilidad histórica de sus precios, lo que a mi parecer puede representar un desafío interesante para los modelos de predicción, y por su importancia económica en México como país productor líder mundial.

**Datos utilizados**

El dataset contiene registros de precios promedio de aguacate desde febrero 2019 hasta 2024, con un total de 85 observaciones después del preprocesamiento. Los precios oscilan entre $35.00 y $118.50 pesos, mostrando una variabilidad considerable que refleja factores estacionales y de mercado.

La preparación de datos incluyó la creación de 13 características preprocesadas:

* Características temporales cíclicas (month\_sin, month\_cos, day\_sin, day\_cos)
* Variables lag (precio\_lag1, precio\_lag2, precio\_lag3)
* Medias móviles (precio\_ma3, precio\_ma6)
* Indicadores de volatilidad y cambios (precio\_std3, precio\_diff1, precio\_diff2)
* Tendencia temporal (days\_since\_start)

**Librerías**

pandas, numpy, scikit-learn, matplotlib, joblib, openpyxl, python-dateutil

módulos de sklearn:

* TimeSeriesSplit: Para validación cruzada que respeta el orden temporal
* MinMaxScaler: Mejor que StandardScaler para SVM, escala entre 0-1
* RandomForestRegressor: modelos de comparación para evaluar si SVM realmente funciona mejor
* mean\_absolute\_error: Métrica adicional más interpretable que MSE

**Ingeniería de características**

* Año: Captura tendencias a largo plazo
* Mes: Estacionalidad mensual (aguacates más caros en invierno)
* Día del año: Estacionalidad más fina
* Trimestre: Patrones trimestrales

**¿Por qué seno y coseno en meses?**

El problema con usar solo month (1,2,3...12) es que el modelo ve diciembre (12) como "muy diferente" de enero (1), cuando en realidad son consecutivos.

* Con seno/coseno:
* Enero: sin(2π×1/12) = 0.5, cos(2π×1/12) = 0.87
* Diciembre: sin(2π×12/12) = 0, cos(2π×12/12) = 1

Estos valores están "cerca" matemáticamente, y eso a su vez representa que enero y diciembre son meses consecutivos.

**¿Por qué lags?**

Los precios de productos agrícolas tienen "memoria": si el aguacate estuvo caro el mes pasado, es probable que siga caro este mes por las mismas condiciones (clima, oferta, etc.).

* lag1: Precio del mes anterior
* lag2: Precio de hace 2 meses
* lag3: Precio de hace 3 meses

**¿Por qué volatilidad?**

La desviación estándar móvil captura qué tan "inestables" han sido los precios. Alta volatilidad puede indicar eventos especiales.

**¿Por qué diferencias?**

diff() calcula el cambio: si el precio subió o bajó respecto al período anterior.

**¿Por qué división temporal y no aleatoria?**

En series temporales no se suele usar train\_test\_split aleatorio porque:

* Sería hacer trampa: usar datos futuros para predecir el pasado
* No reflejaría el uso real: siempre predecimos el futuro basado en el pasado

**¿Por qué MinMaxScaler en lugar de StandardScaler?**

* MinMaxScaler escala entre 0-1
* SVM funciona mejor con datos en rangos similares
* StandardScaler puede generar valores negativos que complican la interpretación

**¿Por qué escalar también y (target)?** Ayuda al SVM a converger más rápido y ser más estable numéricamente.

**¿Cómo funciona TimeSeriesSplit?** En lugar de mezclar los datos, hace divisiones respetando el tiempo:

**¿Qué hace cada hiperparámetro?**

**C**: Controla el trade-off entre ajuste perfecto y simplicidad (el balance)

* C alto: Más complejo, puede sobreajustar
* C bajo: Más simple, puede subajustar

**gamma**: Controla la influencia de cada punto de entrenamiento

* gamma alto: Solo puntos muy cercanos influyen (más complejo)
* gamma bajo: Puntos lejanos también influyen (más simple)

**epsilon**: Define el "tubo" de tolerancia al error

* epsilon alto: Acepta más error, modelo más simple
* epsilon bajo: Menos tolerante al error, más complejo

**kernel**: Función para transformar los datos

* 'rbf': Radial (bueno para patrones complejos)
* 'poly': Polinomial (bueno para relaciones polinómicas)

**¿Qué significa cada métrica?**

* MSE: Error cuadrático medio (penaliza errores grandes)
* RMSE: Raíz del MSE (en las mismas unidades que el precio)
* MAE: Error absoluto medio (más robusto a outliers)
* R²: Porcentaje de variabilidad explicada (0-1, más alto mejor)
* MAPE: Error porcentual medio (fácil de interpretar)

**Modelos entrenados y resultados**

|  |  |  |
| --- | --- | --- |
| Modelos | Hiperparámetros optimizados  Y características empleadas | Métricas de evaluación |
| SVM (Support Vector Machine) | C: 50  epsilon: 0.01  gamma: 0.01  kernel: rbf  características utilizadas: 13  características: days\_since\_start, month\_sin, month\_cos, day\_sin, day\_cos, precio\_lag1, precio\_lag2, precio\_lag3, precio\_ma3, precio\_ma6, precio\_std3, precio\_diff1, precio\_diff2 | SVM:  R² Score: 0.9973  RMSE: 0.72  MAE: 0.59  MAPE: 0.86% |
| Random Forest | características utilizadas: 13  características: days\_since\_start, month\_sin, month\_cos, day\_sin, day\_cos, precio\_lag1, precio\_lag2, precio\_lag3, precio\_ma3, precio\_ma6, precio\_std3, precio\_diff1, precio\_diff2 | Random Forest:  R² Score: 0.9528  RMSE: 3.00  MAE: 2.05  MAPE: 3.07% |
| Perceptrón | mejores hiperparámetros encontrados:  ----------------------------------------  activation: relu  alpha: 0.0001  hidden\_layer\_sizes: (100, 50, 20)  learning\_rate: constant | mean squared error (MSE): 2944.45  R² Score: -17.2071 |
| CNN | información del entrenamiento  ----------------------------------------  epochs\_total: 23  best\_val\_loss: 202.85171508789062  final\_train\_loss: 560.098388671875  lr\_reductions: 0 | mean squared error (MSE): 227.74  R² Score: -0.4082  Epochs: 18 |

**Análisis de los resultados**

Interfaz de usuario gráfica, Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

**Modelos exitosos:**

1. El SVM mostró buen rendimiento con R²=0.9973 y MAPE de solo 0.86%, indicando predicciones precisas. La optimización de hiperparámetros fue efectuada con GridSearchCV con validación temporal.
2. Random Forest, aunque fue inferior al SVM, tiene un buen desempeño con R²=0.9528, también presenta robustez para este tipo de datos.

**Modelos con dificultades:**

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

1. El perceptrón multicapa falló con R²=-17.21, indicando predicciones peores que usar simplemente el promedio. Esto puede deberse a problemas en la arquitectura, o convergencia durante el entrenamiento.

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

1. La CNN también mostró un desempeño pobre (R²=-0.41), siendo redes neuronales profundas (multicapa), puede ser por la cantidad reducida de data, además de la naturaleza de la misma. Esto resalta que la complejidad del modelo no garantiza mejor desempeño, especialmente con datasets pequeños (85 observaciones).

**Reflexiones sobre el desempeño**

Este proyecto demostró varios puntos importantes sobre el machine learning aplicado a series temporales financieras:

* Es importante implementar ingeniería de características previa al entrenamiento del modelo. Las características cíclicas capturaron efectivamente la estacionalidad del aguacate, mientras que los lags y medias móviles proporcionan la "memoria" necesaria para entender tendencias de precios.
* Los modelos tradicionales (SVM, Random Forest) superan en este dataset significativamente a las redes neuronales (en este caso específico).
* Las redes neuronales requieren mucho más datos para entrenar efectivamente. Con solo 85 muestras, estos modelos probablemente no pudieron aprender patrones significativos, resultando en sobreajuste o convergencia pobre.
* La validación temporal es muy importante en series de tiempo. Los buenos resultados del SVM se deben en parte a que se respetó la naturaleza secuencial de los datos durante la validación.

**Para mejorar este análisis se puede optar también por lo siguiente**:

1. Incorporar variables externas como datos climáticos, índices económicos, o eventos estacionales específicos del aguacate.
2. Explorar también otros modelos especializados en series temporales como ARIMA, Prophet, etc.
3. Aumentar el dataset con datos de diferentes regiones o fuentes para robustecer los modelos de deep learning.

Se concluye que este proyecto demuestra que en problemas de predicción de precios con datos limitados, los algoritmos clásicos de machine learning con buena ingeniería de características pueden ser más efectivos que arquitecturas complejas de deep learning.