Maestría en Inteligencia Artificial Aplicada

Materia: Proyecto Integrador

Profesor Titular: Dra. Grettel Barceló Alonso / Dr. Luis Eduardo Falcón Morales

Asesor de Proyecto: Dr. Carlos Alberto Villaseñor Padilla

Avance 5. Modelo Final

Equipo 10

David García Robles A01152606

David Nava Jiménez A01168501

José Antonio Hernández Hernández A01381334

Fecha: 01 de Junio de 2025

Optimización de ventas en Nacional Monte de Piedad

Avance 5. Modelo final

Contenido

Introducción

Es importante mencionar que el dataset con el que vamos a trabajar y el que mostró tener menos ruido es el de **Alhajas**, por lo que el modelo aplicado será a este ramo. A lo largo de esta semana estuvimos trabajando y realizando un análisis a detalle de todas las variables predictoras que consideramos en el modelo. Identificamos que la variable RANGO_DIAS_ALMONEDA, se deriva de nuestra variable objetivo por lo que la eliminamos del modelo y nuestras métricas bajaron conforme a lo presentado en el avance 4. Sin embargo, el modelo final pasó por un proceso de fine-tuning con búsqueda de hiper parámetros óptimos, además de aplicar métodos de ensamble de los dos modelos más robustos que encontramos que fue el XGBoostRegressor, así como el Multi-Layer Perceptron (MLP) dando como resultado un modelo sin sobreajuste y con una métrica de R^2 de .8, lo cual se considera aceptable y satisfactoria, considerando que el dataset original contiene bastante ruido.

Adicionalmente, se presentará una tabla comparativa de los modelos que se ocuparon en el proyecto y cuáles fueron los tiempos de entrenamiento de cada uno, así como el resultado de la métrica R^2 y RMSE (mide en función de las unidades de la variable objetivo-días).

Finalmente, se realizó un modelo con TSNE (reducción de dimensionalidad) como modelo adicional, para agregar más valor a nuestro cliente. Esto nos permite identificar algunos insights ocultos en la varianza entre variables.

1.1 Optimización de hiper parámetros - Mejor modelo

Nuestro mejor modelo fue el XGBoost Regressor, el cual presentó un R^2 de .7 y un RMSE de 67 días, seguido del MLP con un R^2 de .64 y un RMSE de 66 días.

Como primer enfoque decidimos aplicar un RandomSearch al XGBoost Regressor para buscar los mejores hiper parámetros del modelo.

```
param_dist_final_tune_xgboost = {
    'n_estimators': [100,200,300,400],
    'max_depth':[10,20,25,30,50],
    'learning_rate': [0.001,0.01,0.1,0.05,0.2,0.5],
    'subsample':[0.1,0.5,1.0,1.5,2.0],
    'booster':['gbtree','gblinear','dart']
}
xgb_tuned = XGBRegressor(objective='reg:squarederror')

cv = RepeatedKFold(n_splits=2,n_repeats=1)

random_search_final = RandomizedSearchCV(
```

```
    estimator=xgb_tuned,
    param_distributions=param_dist_final_tune_xgboost,
    n_iter=2,
    scoring='r2',
    cv=cv,
    verbose=1,
    n_jobs=1,
    random_state=27
)

print(" Resultados")
random_search_final.fit(Xtrain, ytrain)

                                                                                    #
Resultados
print("\ Mejor modelo ajustado:")
print(f"R2 validación cruzada: {random_search_final.best_score_:.4f}")
print("Mejores hiperparámetros:")
print(random_search_final.best_params_)
```

```
 bst.update(dtrain, iteration=i, fobj=obj)

\ Mejor modelo ajustado:

R2 validación cruzada: 0.5062

Mejores hiperparámetros:

{'subsample': 1.5, 'n_estimators': 100, 'max_depth': 25, 'learning_rate': 0.2, 'booster': 'gblinear'
```

Comentarios: Podemos observar que la búsqueda de hiper parámetros arrojó variables que dieron una R^2 baja. Por esta razón investigamos la forma de mejorar la métrica y fuimos modificando los parámetros para obtener la mejor métrica y cuando el sobre ajuste, estos fueron los mejores hiper parámetros.

```
XGBRegressor(booster='gbtree',n_estimators=330, max_depth=9,learning_rate=0.1,subsample=1.0,objective=
'reg:squarederror',reg_alpha=0.8,reg_lambda=1.0,colsample_bytree=0.9,colsample_bylevel=0.9)
```

Nota: Para el MLP, la cantidad de recursos computacionales aplicados al RandomSearch, fueron excesivos por lo que se decidió mantener los parámetros originales.

1.2 Aplicación de estrategia de ensamble heterogénea

A continuación, aplicamos el modelo Voting Regressor, el cual conjunta un modelo de MLP y XGBoost Regressor, con la finalidad de mejorar nuestra métrica objetivo, así como reducir el sobreajuste que pueda tener un modelo homogéneo.

```
from sklearn.ensemble import VotingRegressor

mlp =
MLPRegressor(hidden_layer_sizes=(700,),activation='relu',max_iter=12,alpha=0.0001,learning_rate='consta
nt')
xgb = XGBRegressor(booster='gbtree',n_estimators=320,
max_depth=9,learning_rate=0.1,subsample=1.0,objective=
'reg:squarederror',reg_alpha=0.8,reg_lambda=1.0,colsample_bytree=0.9,colsample_bylevel=0.9)

voting = VotingRegressor(estimators=[
    ('mlp',mlp),
    ('xgb',xgb)
])

#Configurar validación cruzada
kf = RepeatedKFold(n_splits=5,n_repeats=2)

#Ejecución de validación cruzada
cv_results = cross_validate(voting,
Xtrainval,ytrainval,cv=kf,scoring={'r2':'r2','rmse':'neg_root_mean_squared_error'},return_train_score=T
rue)

print(f"\tTrain R2: {np.mean(cv_results['train_r2']):.3f} ± {np.std(cv_results['train_r2']):.3f}")
print(f"\tVal   R2: {np.mean(cv_results['test_r2']):.3f} ± {np.std(cv_results['test_r2']):.3f}")
print(f"\tTrain RMSE: {-np.mean(cv_results['train_rmse']):.3f} ±
{np.std(cv_results['train_rmse']):.3f}")
print(f"\tVal RMSE: {-np.mean(cv_results['test_rmse']):.3f} ± {np.std(cv_results['test_rmse']):.3f}")
```

```
Train R2: 0.819 ± 0.001
      Val   R2: 0.778 ± 0.003
      Train RMSE: 48.286 ± 0.113
      Val RMSE: 53.555 ± 0.552
```

Interpretaciones

El modelo tardó en entrenarse 4,810 seg, el más alto entre todos los modelos y con un resultado similar al XGBoost, en cuanto a las dos métricas que estamos utilizando que son R^2 0.78 y un RMSE de 53.6 días.

1.3 Tabla comparativa de los Modelos de Machine Learning y el método heterogéneo evaluado

Realizando la comparación entre los diferentes modelos obtuvimos los siguientes resultados, los cuales se encuentran ordenados de mejor desempeño, a más bajo.

Modelo	Tiempo entrenamiento (seg)	Métrica R^2	Métrica RMSE
XGBoost Regressor	780	0.79	51.9
Voting Regressor	4810	0.78	53.6

MLP	2,470	0.72	59.7
LinearRegression	20	0.52	78.1
ElasticNet	545	0.49	80.3
Decission Tree	37	0.46	83.3
Random Forest	46	0.26	97.9

Comentarios:

En la comparación de modelos utilizando validación cruzada, se evaluaron seis algoritmos: ElasticNet, Árbol de Decisión, Random Forest, XGBoost, Red neuronal de perceptrón multicapa y Máquina de Vector Soporte. Podemos observar que el modelo con mayor desempeño es el **XGBoost Regressor con .79 de R^2 y 51.96 RMSE.**

Comentando con NMP, su expectativa considera que los datos compartidos tienen bastante ruido, su consideración en cuanto a la desviación del modelo se comentó entre 35-45 días, por lo que consideramos que

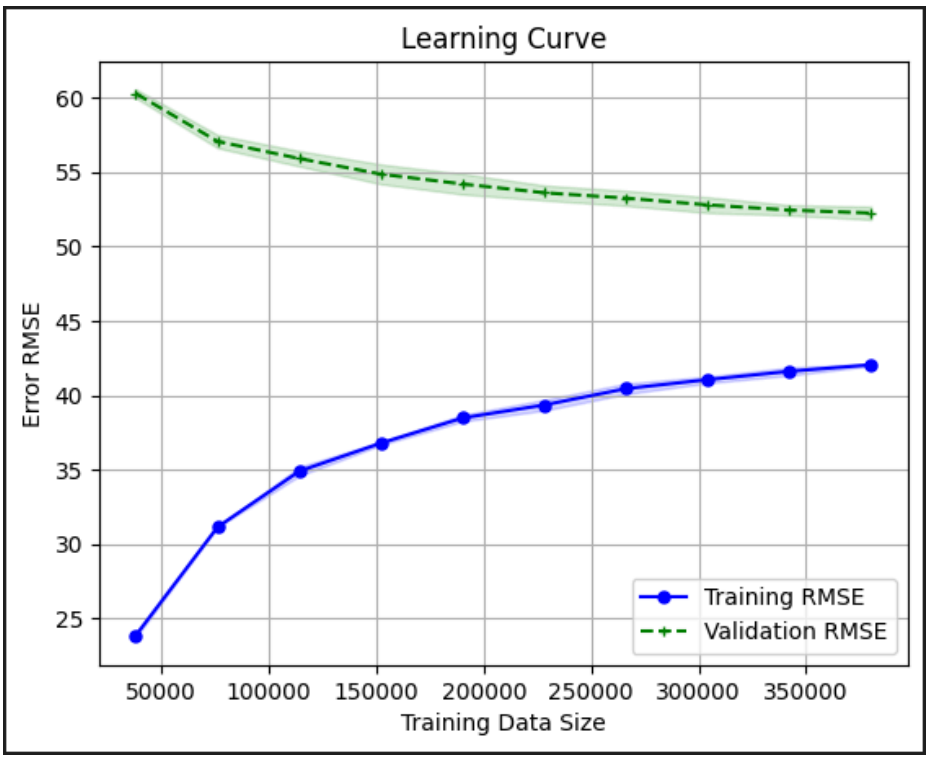
1.4 Gráficos y visualizaciones del mejor modelo

Se realizó un gráfico evaluando el desempeño de la métrica RMSE (root mean squared error) con los datos de entrenamiento y prueba a lo largo de las iteraciones del entrenamiento, la finalidad es evaluar el rendimiento del modelo y visualizar si existe sobreajuste o sub ajuste.

En nuestro caso, graficamos el modelo XGBoost Regressor.

```
best_model = XGBRegressor(booster='gbtree',n_estimators=320,
max_depth=9,learning_rate=0.1,subsample=1.0,objective=
'reg:squarederror',reg_alpha=0.8,reg_lambda=1.0,colsample_bytree=0.9,colsample_bylevel=0.9)

train_sizes, train_scores,val_scores = learning_curve(estimator=best_model,
                                                      X=Xtrain,
                                                      y = ytrain,
                                                      cv= 5,
                                                      train_sizes= np.linspace(0.1,1.0,10),
                                                      scoring= 'neg_mean_squared_error',
                                                      n_jobs=1)
rmse_train_scores = np.sqrt(-train_scores)
rmse_val_scores = np.sqrt(-val_scores)

train_mean = np.mean(rmse_train_scores, axis=1)
train_std = np.std(rmse_train_scores, axis=1)
val_mean = np.mean(rmse_val_scores, axis=1)
val_std = np.std(rmse_val_scores, axis=1)
```

```
plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5, label='Training RMSE')
plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std, alpha=0.15, color='blue')

plt.plot(train_sizes, val_mean, color='green', marker='+', markersize=5, linestyle='--',
label='Validation RMSE')
plt.fill_between(train_sizes, val_mean + val_std, val_mean - val_std, alpha=0.15, color='green')

plt.title('Learning Curve')
plt.xlabel('Training Data Size')
plt.ylabel('Error RMSE')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```

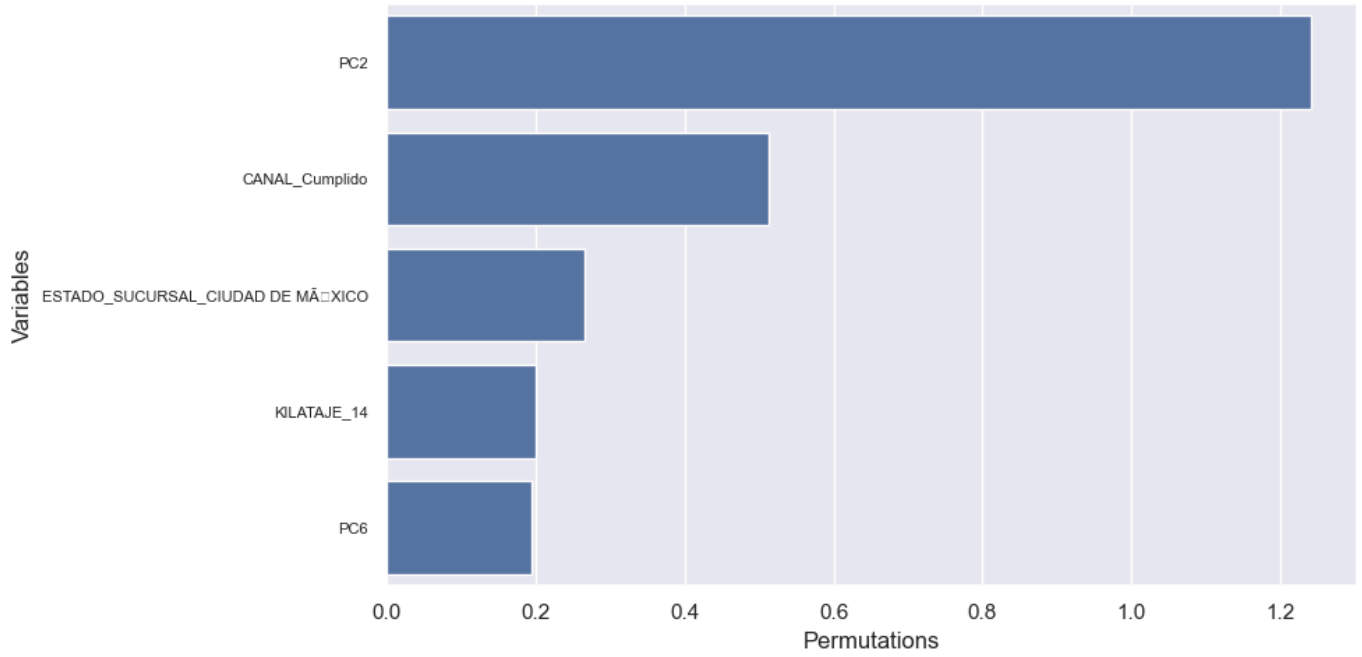


Podemos observar que conforme el tamaño de datos de entrenamiento aumenta, las curvas tienden a converger, podemos también observar que existe un ligero sobreajuste del modelo, el cual no es significativo y podemos ocuparlo como nuestro modelo final.

Por otro lado, realizamos un **análisis de la importancia de variables** en nuestro modelo y a su vez predecir nuestra variable DIAS_ALMONEDA.

```
from sklearn.inspection import permutation_importance
results = permutation_importance(xgb,Xtest,ytest,scoring ='r2')
importance_per = results.importances_mean
```

```
sns.set(rc={'figure.figsize':(10,5)})
sns.barplot(y='Variables',x='Permutations',data=permutation_df.iloc[0:5],errorbar=None)
plt.yticks(fontsize=8)
plt.tight_layout()
```



Podemos observar que las 5 variables que más influyen en nuestro modelo de acuerdo al análisis de permutaciones son PC2-Des_EXT, Canal-Cumplido, Estado_sucursal_Ciudad_de_México, Kilataje_14, PC6- Interés.

1.5 Modelo final aplicado al set de datos de prueba

Tras una evaluación rigurosa de múltiples algoritmos de regresión, se definió como modelo final una estrategia de ensamble heterogénea basada en un VotingRegressor, el cual integra dos estimadores complementarios: un XGBoostRegressor y un MLPRegressor. Esta configuración permite capturar tanto relaciones lineales como patrones no lineales complejos presentes en los datos, maximizando así la capacidad predictiva del modelo. Lo consideramos el mejor modelo debido a dos características, desempeño y el más bajo sobre ajuste de todos los modelos aunque el costo computacional es el más alto de los modelos.

Train R2: 0.819 ± 0.001

Val R2: 0.778 ± 0.003

Train RMSE: 48.286 ± 0.113

Val RMSE: 53.555 ± 0.552

El modelo ajustado mediante una búsqueda aleatoria de hiperparámetros RandomizedSearch, logró un desempeño sobresaliente durante la validación cruzada, con un coeficiente de

determinación R2 de 0.78 y un error cuadrático medio (RMSE) de 53.6. Estos resultados evidencian una capacidad robusta para generalizar a nuevos datos.
En paralelo, se implementó una versión optimizada del mismo ensamble, sin escalamiento adicional y utilizando los hiperparámetros previamente identificados. Esta versión fue diseñada con el objetivo de mejorar la eficiencia computacional sin comprometer de forma significativa la precisión del modelo. Los resultados obtenidos muestran un R2 promedio de 0.678 y un RMSE de 61.65, con un tiempo de entrenamiento considerablemente menor. Si bien su desempeño fue ligeramente inferior al del modelo principal, sus características lo hacen atractivo para escenarios donde los recursos computacionales son limitados o se requiere una mayor rapidez en el despliegue.
Ambas configuraciones demostraron ser consistentes al aplicarse sobre el conjunto de datos de prueba, validando su utilidad práctica para anticipar el tiempo que una alhaja permanecerá en inventario. Esta capacidad predictiva puede ser integrada estratégicamente por Nacional Monte de Piedad para optimizar la gestión de inventarios, ajustar políticas de precios, y mejorar la toma de decisiones operativas en sus sucursales.

Conclusiones

A lo largo del desarrollo del modelo de predicción para estimar los días que una prenda permanecerá en almoneda, se implementaron diversas estrategias de modelado y validación con el objetivo de maximizar la capacidad explicativa del modelo sin comprometer su generalización.

Durante las primeras fases, se observó que el uso de la variable RANGO_ALMONEDA proporcionaba un desempeño excepcional en términos de R2, alcanzando valores cercanos al 0.90. Sin embargo, tras un análisis detallado, se concluyó que esta variable tenía una alta correlación con la variable objetivo, lo cual introducía un sesgo significativo y un riesgo de data leakage. Por esta razón, se optó por eliminarla del conjunto de datos final, resultando en un desempeño más conservador pero representativo del comportamiento esperado en producción.

En cuanto a los modelos implementados, el enfoque que ofreció los mejores resultados fue el Voting Regressor, un modelo heterogéneo que combina las predicciones de XGBoostRegressor

y MLPRegressor. Este método alcanzó un R2 de 0.78 y un RMSE de 53.6 días, manteniéndose muy cercano al mejor modelo individual (XGBoost) que obtuvo R2 de 0.79 y RMSE de 51.9 días. Lo anterior demuestra que los modelos de ensamblado pueden representar una solución sólida y balanceada para este tipo de problemas, permitiendo captar tanto relaciones lineales como no lineales en los datos.

Por otro lado, se exploró la implementación de un modelo StackingRegressor, que teóricamente podría ofrecer mejoras adicionales al aprender a combinar las fortalezas de distintos modelos base. Sin embargo, la complejidad computacional asociada al entrenamiento cruzado de modelos como MLPRegressor dentro del stacking provocó tiempos de ejecución excesivos, dificultando su aplicación efectiva en este ciclo de desarrollo. Por ello, aunque se reconoce el potencial del stacking, se sugiere su implementación futura en un entorno con mayores recursos computacionales o mediante la optimización del diseño del modelo.

En resumen, el proceso permitió identificar la importancia de eliminar variables con alto poder explicativo pero con potencial de fuga de información, además de confirmar que los modelos heterogéneos como VotingRegressor son una estrategia viable para lograr un desempeño competitivo sin comprometer la estabilidad ni la interpretabilidad. Estos hallazgos sientan una base sólida para futuras mejoras y despliegues en entornos reales.