ENTREGA FINAL PROYECTO INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

ISABELA PAREJA GIL

JULIÁN DAVID OLAYA RESTREPO

ESTEBAN CARO PELÁEZ

DOCENTE INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

Raúl Ramos Pollán

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

MEDELLÍN -ANTIOQUIA

Tabla de contenido

Introducción

El presente trabajo se refiere a la implementación de un algoritmo de clasificación para el problema planteado en la competición "*Don't Get Kicked!*" en Kaggle. La cuestión principal de este trabajo se enfoca en predecir si un concesionario va a realizar una mala compra teniendo en cuenta diferentes variables recolectadas previamente acerca del auto a adquirir.

1. Descripción del problema predictivo

Una de las formas de adquisición de vehículos en los concesionarios es mediante la compra de automóviles en subasta. Muchos de estos vehículos tienen fallas mecánicas recurrentes que pueden estar relacionadas con diferentes factores; problemas de autenticación del vendedor, modificación del tacómetro para el kilometraje, fallas mecánicas, entre otros. La compra de dichos vehículos llegan a ser inversiones riesgosas, puesto que en muchas ocasiones, la cantidad de dinero necesario para solventar dichas fallas superan la ganancia esperada por el concesionario.

1.1 Descripción del dataset

El dataset a utilizar fue obtenido de la competición en Kaggle "Don't Get Kicked!".Cuenta con 121.692 registros de autos, de los cuales el 60% es destinado para el training y el 40% para testing. El dataset cuenta con el 40% de los datos categóricos y el 60% numéricos.

1.2 Descripción de las variables

RefId: identificador del vehículo dentro de la base de datos
IsBadBuy: variable binaria que representa si el vehículo fue una mala compra (1) o no (0).
PurchDate: fecha en la que se compró el vehículo en la subasta
Auction: proveedor a quien se le compró el vehículo en la subasta
VehYear: año de fabricación del vehículo
VehicleAge: años transcurridos desde el año de fabricación del vehículo
Make: marca del fabricante del vehículo
Model: modelo del vehículo
Trim: versión del modelo con una configuración particular
SubModel: submodelo del vehículo
Color: color del vehículo
Transmission: tipo de transmisión del vehículo (automática o manual)
WheelTypeID: identificador del tipo de llanta (Covers 2 - Alloy 1)
WheelType: tipo de llanta (Covers, Alloy)
VehOdo: kilometraje del vehículo
Nationality: nacionalidad del fabricante del vehículo
Size: categoría del tamaño del vehículo
TopThreeAmericanName: identifica si el vehículo hace parte del top 3 de fabricantes estadounidenses.
WarrantyCost: El precio de la garantía con un términos de 36 meses y y 36 mil millas.
MMRAcquisitionAuctionAveragePrice: Precio de adquisición para este vehículo en estado medio en el momento de la compra

MMRAcquisitionAuctionCleanPrice: Precio de adquisición para este vehículo en un estado superior a la media en el momento de la compra
MMRAcquisitionRetailAveragePrice: Precio de adquisición de este vehículo en el mercado minorista en estado medio en el momento de la compra
MMRAcquisitonRetailCleanPrice: Precio de adquisición de este vehículo en el mercado minorista en un estado superior a la media en el momento de la compra.
MMRCurrentAuctionAveragePrice: Precio de adquisición de este vehículo en estado medio en el día actual
MMRCurrentAuctionCleanPrice: Precio de adquisición de este vehículo en el estado anterior en el día actual
MMRCurrentRetailAveragePrice: Precio de adquisición de este vehículo en el mercado minorista en estado medio en el día actual.
MMRCurrentRetailCleanPrice: Precio de adquisición para este vehículo en el mercado minorista en condiciones superiores a la media en el día actual.
PrimeUnit: Identifica si el vehículo tendría una demanda mayor que una compra estándar
AcquisitionType: Identifica cómo se adquirió el vehículo (subasta, intercambio, etc.).
AUCGUART: El nivel de garantía proporcionado por la subasta para el vehículo (Luz verde - Garantizado/interceder, Luz amarilla - precaución/problema, luz roja - vendido tal cual)
KickDate: Fecha de devolución del vehículo a la subasta
BYRNO: Número único asignado al comprador que adquirió el vehículo
VNZIP: Código postal donde se compró el automóvil
VNST: Estado en el cual el automóvil fue comprado
VehBCost: Costo de adquisición pagado por el vehículo en el momento de la compra
IsOnLineSale: identifica si el vehículo se compró originalmente en línea.

1.3 Métrica de machine learning

Como métrica de desempeño de machine learning vamos a utilizar el Accuracy dado que es un problema de clasificación binaria (1 y 0).

$$Accuracy = \frac{Number\ of\ correct\ prediction}{Total\ number\ of\ predictions}$$

1.4 Métrica de negocio y desempeño deseable en producción

Se determina el *margen de ganancia* neta como métrica de negocio, el cual se calcula mediante el ingreso neto (ganancia después de gastos de mantenimiento al auto) dividido entre el precio de venta al futuro comprador para cada auto comprado basados en el algoritmo predictivo.

Se medirá el desempeño del modelo teniendo en cuenta el porcentaje de ganancia (margen de ganancia neta) por cada vehículo vendido. Cuando éste porcentaje sea mayor a un 15% se considera como una compra viable.

2. Análisis exploratorio (DEA)

El dataset tiene 72983 filas y 34 columnas.

2.1 Variable objetivo

Se realizó un histograma para inspeccionar la variable objetivo, el cual permite identificar con facilidad al ser esta una variable binaria. Se observa que aproximadamente el 12% de las compras fueron una mala decisión (1) y la mayoría fueron buena decisión (0)

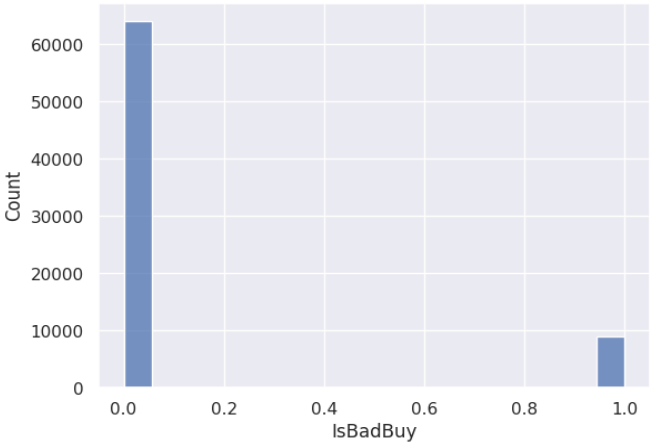

Figura 1. Distribución de la variable objetivo

2.2 Correlación entre variables numéricas

Se grafica una matriz de correlación entre variables.

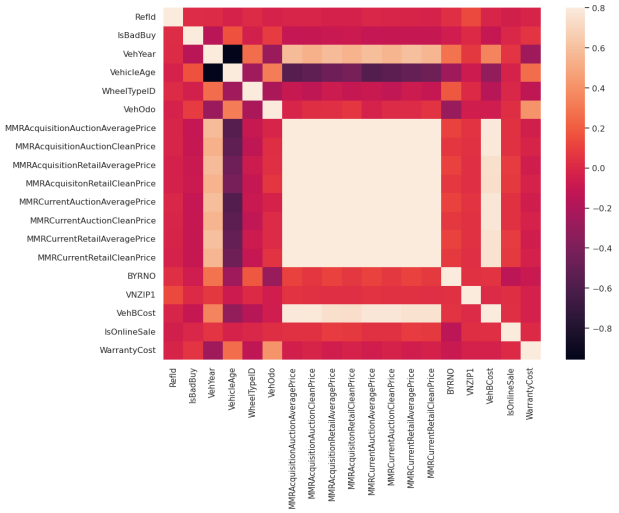

Figura 2. Matriz de correlación entre variables

Las variables que mayor correlación presentan son: la de VehYear y VehicleAge, ya que mientras una va aumentando en año, la otra va disminuyendo en edad. Las variables MMRA tienen correlación positiva entre ellas mismas.

2.3 Datos faltantes

El 6% son datos faltantes (de 2.481.558 datos, 149.185 son faltantes). Se realiza el gráfico de datos faltantes para identificar aquellas columnas en las cuales se debe tomar una decisión de eliminación.

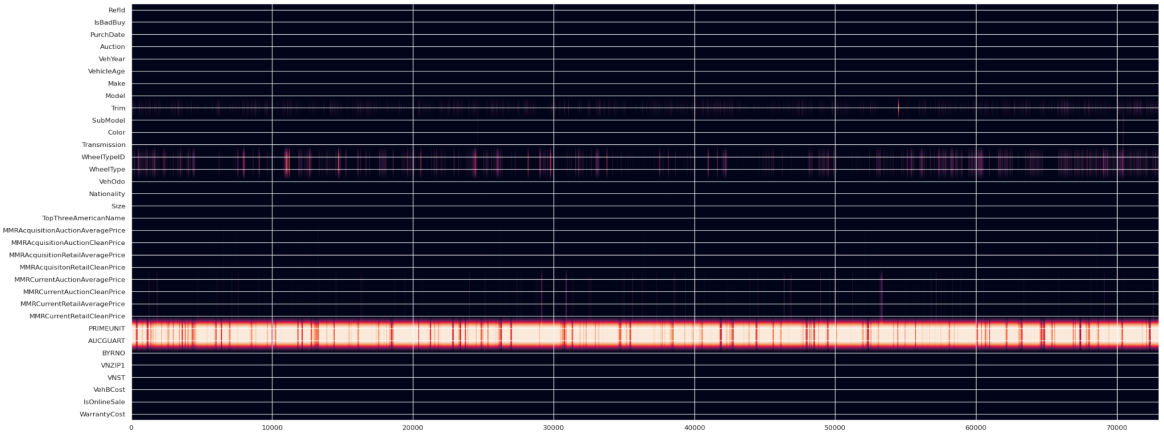

Figura 3. Visualización de datos faltantes

Nótese que hay una franja blanca en la figura 3, la cual corresponde a las variables PRIMEUNIT y AUCGUART.

2.4 Datos categóricos

2.4.1 Lugar de fabricación del vehículo

Se observa que en el dataset, los vehículos adquiridos son mayormente producidos por marcas norteamericanas, seguido de vehículos asiáticos.

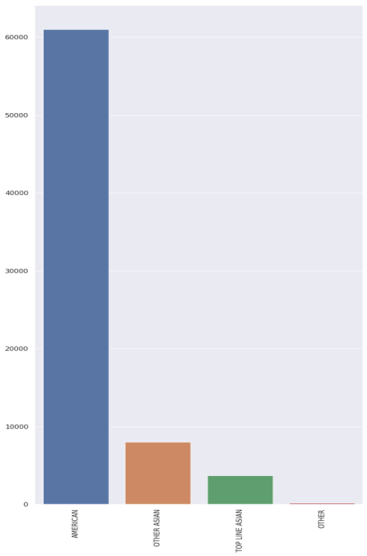

Figura 3. Histograma lugar de fabricación

2.4.2 Tamaño del vehículo

En el dataset, la mayoría de los vehículos son de tamaño mediano y hay muy pocos autos deportivos en nuestras muestras.

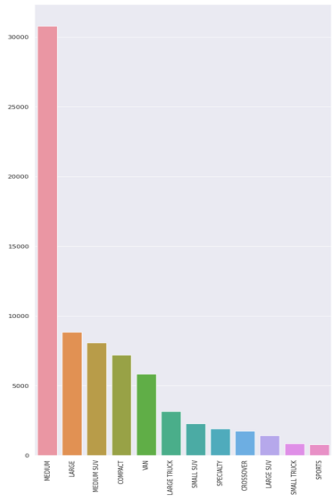

Figura 4. Histograma tamaño del vehículo

3. Tratamiento de datos

3.1 Eliminación de columnas

- PRIMEUNIT,AUCGUART: se eliminan dada la cantidad de datos faltantes que ellas tienen como se observó el gráfico de nulos.
- Wheel Type: nos da la misma información de WheelTypeID
- BYRNO: número asignado al comprador del vehículo, no es importante en nuestro estudio
- Veh Year es similar a Vehicle Age
- Por el principio de parsimonia, eliminamos Trim, Model y SubModel
- Tampoco necesitamos RefId
- PurchDate no nos provee información valiosa.
- VNST y VNZIP: Estado y código postal del vehículo, lo cual tampoco provee información importante para la determinación de una buena o mala compra.

3.2 Imputación de variables numéricas

Para los datos faltantes en las variables numéricas, en la mayoría se decidió hallar la media de cada variable, para así permitir que los datos sigan su patrón de comportamiento. Utilizando la siguiente estructura de código para cada variable:

```
1 warranty_mean = round(d1['WarrantyCost'].mean(),2)
2 d1.loc[d1['WarrantyCost'].isna(),'WarrantyCost'] = warranty_mean
```

Figura 5. Código implementado para tratamiento de variables numéricas

3.3 Imputación de variables categóricas

Haremos uso de la imputación de la moda que consiste en el reemplazo de valores faltantes por los valores de la moda en cada columna, es decir, la observación más común. Utilizando la siguiente estructura de código para cada variable:

```
moda_nationality = d1['Nationality'].value_counts().idxmax()
d1.loc[d1['Nationality'].isna(),'Nationality'] = moda_nationality
```

Figura 6. Código implementado para tratamiento de variables categóricas

3.4 Transformación de variables categóricas

Después de identificadas y tratadas las variables categóricas, se utiliza el siguiente bloque de código para transformarse y ser utilizadas en el modelo:

```
dftrain= pd.get_dummies(d1)
dftrain
```

Figura 7. Código implementado para transformación de variables categóricas

4. Métodos supervisados

4.1 Métrica de machine learning

Como se mencionó anteriormente, la métrica de machine learning a utilizar será accuracy dado que se está ante un problema de clasificación.

$$Accuracy = \frac{Number\ of\ correct\ prediction}{Total\ number\ of\ predictions}$$

4.2 Partición de los datos

Para el entrenamiento de los modelos se usa el dataset X_train, al cual se le realiza una partición en dos partes. La primera, llamada Xtv se utiliza para entrenar el modelo, y la segunda, Xts, se utiliza para el testeo del modelo.

```
X_train = dftrain.drop([ 'IsBadBuy'], axis=1)
y_train = dftrain['IsBadBuy']
```

Figura 8. Partición de los datos

4.3 Selección de modelos

Dado que estamos ante un problema de clasificación, seleccionamos los algoritmos de Regresión Logística y Árbol de Decisión.

4.4 Regresión Logística

Al intentar calibrar el modelo con los datos mencionados anteriormente, se obtiene una advertencia por parte de Python, así:

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

Figura 9. Mensaje de advertencia de Python en Regresión Logística

Lo anterior está advirtiendo que el algoritmo de optimización detrás de la regresión logística está intentando encontrar los coeficientes que permitan un óptimo ajuste de los datos y no ha logrado encontrarlos sin superar el número máximo de iteraciones. Este problema está relacionado con el **solver** que tiene por defecto la regresión logística, que es un **hiperparámetro** que habrá que modificar para solucionar esto.

```
1 from sklearn.linear_model import LogisticRegression
2 from sklearn.metrics import accuracy_score
3
4 lr = LogisticRegression()
5 lr.fit(Xtv,ytv)
6 lr.score(Xtv,ytv)
7 y_pred = lr.predict(Xtv)
8 acc = accuracy_score(y_pred,ytv)
```

Figura 10. Regresión logística

A continuación realizaremos algunas modificaciones a los hiperparametros del algoritmo para evaluar cúal son mejores.

4.4.1 Mejores hiperparámetros para la regresión logística

Para la definición de los mejores hiperparámetros para el algoritmo hacemos uso de la librería Grid Search CV de Sklearn. En este caso, se va a prestar especial atención a los hiperparámetros Penalty, C y Solver. Esto lo hacemos mediante el siguiente bloque de código:

```
logreg = LogisticRegression()

param_grid = {
    'penalty': ['l1', 'l2'],
    'C': [0.1, 1, 10],
    'solver': ['liblinear', 'saga']
}

grid_search = GridSearchCV(estimator=logreg, param_grid=param_grid, cv=5)

grid_search.fit(Xtv, ytv)
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
```

Figura 11. Búsqueda de los mejores hiperparámetros regresión lineal

Obteniendo los siguientes hiperparámetros, así como el accuracy que se obtendría al implementar dicho algoritmo y con los hiperparametros establecidos:

```
Mejores hiperparámetros: {'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}
Mejor puntaje:0.8767029448058746
```

Finalmente, con la implementación de los parámetros anteriores obtendremos un accuracy del `87.6703%`.

4.5 Árbol de decisión

Los árboles de decisión son un tipo de algoritmo supervisado principalmente utilizado en problemas de clasificación. En este proyecto implementamos dicho algoritmo para generar predicciones con respecto a la variable objetivo.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

decision_tree = DecisionTreeClassifier()
decision_tree.fit(Xtv, ytv)
y_pred = decision_tree.predict(Xtv)
```

Figura 12. Árbol de decisión

Al calcular el accuracy con este algoritmo se obtiene un valor 1. Inicialmente, se podría pensar que el modelo está realizando predicciones completamente acertadas. Ahora bien, es necesario tener en cuenta el contexto propio de nuestros datos. En la figura 1 se observa que en la distribución de la variable objetivo hay un desequilibrio de clases, por lo cual este algoritmo puede estar presentando un sesgo hacia la clase mayoritaria y tener dificultades para predecir correctamente la clase minoritaria.

4.5.1 Mejores hiperparámetros para el árbol de decisión

Ante la situación presentada en el numeral anterior y la incertidumbre de que el modelo esté realizando predicciones de manera correcta tanto para las muestras mayoritarias como para las minoritarias, se implementa la librería GridSearchCV para definir los mejores hiperparámetros para nuestro set de datos.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

decision_tree = DecisionTreeClassifier()
# Defini los hiperparámetros a probar
parameters = {
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
}
grid_search = GridSearchCV(decision_tree, parameters, cv=5)
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
best_score = grid_search.best_score_
print(best_params)
print(best_score)
```

Figura 13. Búsqueda de los mejores hiperparámetros para árbol de decisión

Con el código anterior se encuentran los siguientes hiperparámetros:

```
{'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 2}
0.897935438293152
```

Observamos que el GridSearchV encuentra unos hiperparametros que nos generan un accuracy de cerca del 87,72 % y no del 100%, siendo más confiable estos resultados.

5. Métodos no supervisados

5.1 PCA + Regresión Logística

En este caso se realizó una reducción de la dimensionalidad mediante PCA teniendo en cuenta diferentes números de componentes, para posteriormente implementar una regresión logística que generase las predicciones binarias. Para la implementación de estos algoritmos no supervisados + supervisados decidimos tener en cuenta un hiperparámetro no contemplado en los algoritmos predictivos. Dicho hiperparámetro fue `class_weight="balanced"`. La razón para tener en cuenta dicho hiperparámetro es el gran desbalance de clases que se observa en la Figura 1, en la cual es evidente que se tienen muchos datos de autos que son buenas compras y pocos de autos que son malas compras. El código para determinar el número de componentes óptimos para un PCA, contemplando el hiperparámetro class_weight se observa a continuación:

```python
from sklearn.decomposition import PCA
components = [1,2,3,5,7,9]
test_size = 0.3
val_size  = test_size/(1-test_size)
perf = []
lr = LogisticRegression(class_weight='balanced')
for i in components:
    pca = PCA(n_components = i)
    X_t = pca.fit_transform(X_train)
    Xtv, Xts, ytv, yts = train_test_split(X_t, y_train, test_size=test_size)

    lr.fit(Xtv, ytv)
    perf.append(accuracy_score(yts , lr.predict(Xts)))
```

Figura 14. Número de componentes para PCA + Regresión Logística

El resultado del código anterior es el siguiente:

```
Accuracy del modelo con  1 elementos:  0.48256
------------
Accuracy del modelo con  2 elementos:  0.54212
------------
Accuracy del modelo con  3 elementos:  0.54013
------------
Accuracy del modelo con  5 elementos:  0.53837
------------
Accuracy del modelo con  7 elementos:  0.53402
------------
Accuracy del modelo con  9 elementos:  0.53014
------------
```

Figura 15. Número de componentes para PCA + Regresión Logística

Determinamos que el número de elementos óptimo para la implementación del algoritmo es de 5.

Después de realizar tres iteraciones modificando los hiperparámetos `solver` y `max_iter` concluimos que los mejores hiperparámetros son:
`max_iter=120,solver='newton-cholesky',class_weight='balanced'`

Generándonos un accuracy de 58,65%

5.2 NMF + Árbol de decisión

De manera similar al procedimiento anterior, implementamos el algoritmo NMF para descomponer nuestra matriz de entrenamiento y buscar el número de elementos óptimo para obtener el mejor accuracy al implementar un árbol de decisión después de haber reducido el tamaño de la matriz de entrenamiento. Cabe resaltar que para este caso, se incluyó un hiperparámetro que no se había incluido anteriormente, `class_weight = 'balanced'`. Dicho parámetro tiene en cuenta el desbalance de clases que se observa en la variable objetivo de la figura 1 .

```
Mejor puntaje del modelo con  1 elementos:  0.59744
------------
Mejor puntaje del modelo con  3 elementos:  0.51887
------------
Mejor puntaje del modelo con  5 elementos:  0.60972
------------
Mejor puntaje del modelo con  7 elementos:  0.71600
------------
Mejor puntaje del modelo con  9 elementos:  0.57582
------------
```

Figura 16. NMF + Árbol de decisión

Nótese que hubo una disminución en los puntajes en comparación con los algoritmos anteriores, se observa que el número de elementos óptimos para la descomposición de la matriz es 7.

```python
Xtv, xts, ytv, yts = train_test_split(X_train, y_train, test_size=test_size)

nmf = NMF(n_components = 7)
X_t1 = nmf.fit_transform(Xtv)
Xtv, Xts, ytv, yts = train_test_split(X_t1, ytv, test_size=test_size)
# Crea una instancia de PCA
nmf = NMF(n_components=3)
Des_tree = DecisionTreeClassifier(class_weight='balanced',max_leaf_nodes=10,max_depth=15)
pipeline = Pipeline([('nmf', nmf), ('Decission Classifier', Des_tree)])
pipeline.fit(Xtv, ytv)
y_pred = pipeline.predict(Xtv)

accuracy = accuracy_score(y_pred,ytv)
print(accuracy)
```

Figura 17. NMF + Árbol de decisión

Después de realizar 5 iteraciones con el algoritmo anterior, se llega a la conclusión que los mejores hiperparámetros para este algoritmo son:

`class_weight = 'balanced'`

`max_leaf_nodes = 10`

`max_depth = 15`

6. Curvas de aprendizaje

6.1 Regresión logística

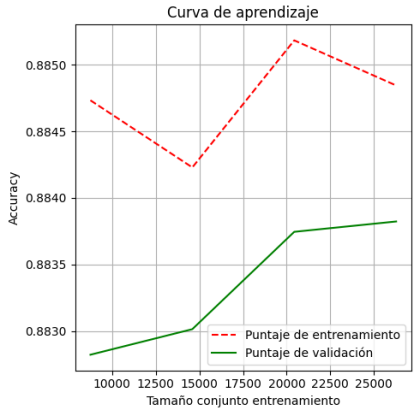

Figura 18. Curva de aprendizaje-Regresión Logística

La figura 18 muestra al inicio de la implementación del modelo, cuando existen pocos datos, una disparidad con respecto a los resultados en entrenamiento y validación. Al inicio de la implementación del modelo, se observa poca precisión en los datos arrojados por la validación, esto debido a la poca densidad de datos disponibles, sin embargo, se observa un aumento en la precisión a medida que el modelo recibe mayor densidad de datos, mostrando una tendencia a la estabilidad. Los datos de entrenamiento, muestran un comportamiento fluctuante, alejándose y acercándose al objetivo del modelo, lo que evidencia la necesidad de ajustar el modelo. Con respecto al comportamiento de los datos obtenidos durante el entrenamiento y la validación, podemos observar una disparidad continua durante la implementación del modelo, lo que evidencia un Overfitting, que posiblemente, con respecto a la naturaleza de los datos del estudio, posiblemente se deban a una mayor complejidad del modelo.

Recomendación para mejoría del desempeño del modelo: en la implementación inicial de este modelo de regresión logística no se contempló el desbalance de clases que se tiene para la variable objetivo, y por dicha razón, a pesar de que los accuracies son muy altos y aparentemente el modelo es muy funcional, la realidad es que al ver otras métricas como F1-Score, esto puede cambiar. Se recomienda contemplar dicho hiperparámetro para el modelo en producción.

6.2 Árbol de decisión

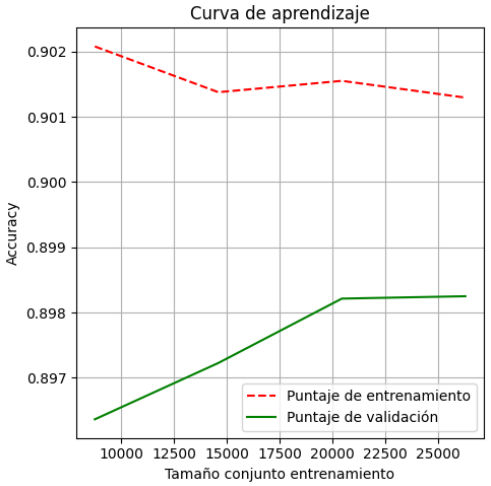

Figura 19. Curva de aprendizaje-Árbol de decisión

En la figura 19, se aprecia al inicio de la implementación del modelo, que los datos arrojados por el entrenamiento presentan gran acercamiento con el objetivo del estudio, y posteriormente un descenso de la precisión o el objetivo esperado. Los datos de la validación por el contrario, se alejan del objetivo de la métrica, y posteriormente, al aumentar la densidad de los datos, se acerca al objetivo de la métrica establecida si observa una tendencia hacia la estabilidad el comportamiento de la validación, sin embargo, no se aprecia una convergencia entre los datos de entrenamiento y validación, lo que evidencia la presencia de un overfitting que se ha mencionado anteriormente.

Recomendación para mejoría del desempeño del modelo: como se ha visto a lo largo del presente informe, el desbalance de clases que se tiene, genera que nuestro modelo se incline a predecir mucho más la variable con la cual contamos con más muestras. Para este caso, se recomienda explorar la posibilidad de obtener más datos acerca de los automóviles que son malas compras siempre y cuando los costos asociados a dicho estudio sea lo suficientemente rentable teniendo en cuenta los costos que implicaría el despliegue del modelo predictivo.

6.3 PCA + Regresión logística

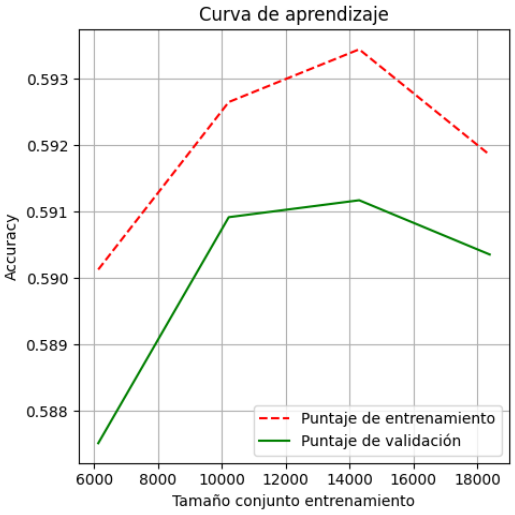

Figura 20. Curva de aprendizaje-PCA+Regresión logística.

En la figura 20, se aprecia un comportamiento similar entre los datos obtenidos por el entrenamiento del modelo y los datos arrojados por la validación, en los cuales se aprecia al inicio de la implementación, que ambos datos comienzan alejados del objetivo de la métrica, y a medida de que aumenta la densidad de los datos, ambos, muestran un comportamiento de ascenso hacia el objetivo de la métrica del modelo y obtienen, tanto los datos de entrenamiento como de validación, un acercamiento máximo cuando existe gran densidad de datos pero mostrando un descenso de la precisión cuando los datos han alcanzado este tope máximo, mostrando a su vez, una tendencia a alejarse del objetivo de la métrica del modelo. Con respecto a la convergencia, se aprecia que estos no convergen o no muestran una tendencia a la convergencia, apreciándose un overfitting para el modelo.

Recomendación para mejoría del desempeño del modelo: en este modelo no se contempló el hiperparámetro de penalty para la regresión logística. Dicho hiperparámetro es utilizado para controlar la regularización dentro de una regresión logística. En particular lo que se busca es resolver el problema conocido como overfitting.

6.4 NMF + Árbol de decisión

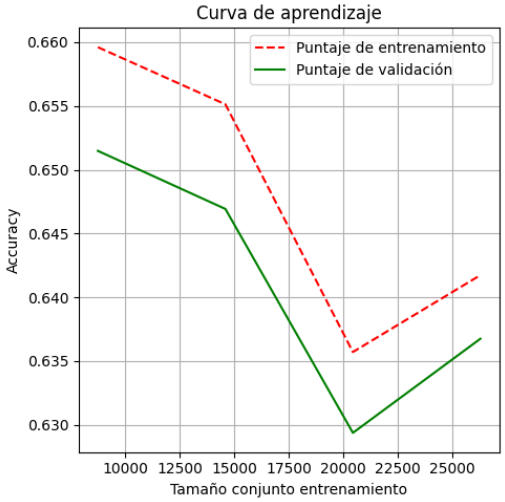

Figura 21. Curva de aprendizaje-NMF+Árbol de decisión.

En la figura 21 se observa un comportamiento similar entre los datos arrojados por el entrenamiento y la validación pero con diferencias en la precisión y acercamiento al objetivo de la métrica. Ambos conjuntos se alejan considerablemente del objetivo conforme a la métrica del modelo. Ambos conjuntos de datos muestran un descenso en la precisión y acercamiento respecto a la métrica del modelo y llegan ambos conjuntos a un tope mínimo para posteriormente, mostrar un marcado ascenso hacia la métrica del modelo. Es evidente el comportamiento de overfitting y no convergencia entre los conjuntos de entrenamiento y validación, sin mostrar una tendencia hacia la convergencia.

Recomendación para mejoría del desempeño del modelo: una posible solución para evitar el overfitting que se está presenciando en el algoritmo NMF + Decision Tree Classifier es disminuir el hiperparámetro max_depth y modificar el hiperparámetro min_samples_split, realizando un proceso iterativo con GridSearchV para la detección de los valores más óptimos.

7. Retos y condiciones de despliegue del modelo

Para evaluar el despliegue en producción de un modelo de predicción sobre la compra de vehículos, es necesario considerar diversos retos y condiciones. Estos aspectos son fundamentales para asegurar que el modelo funcione de manera efectiva y genere valor en el entorno de producción.

Nivel de desempeño mínimo: Para determinar el nivel de desempeño mínimo necesario para desplegar el modelo en producción, se deben establecer criterios basados en los objetivos y requisitos del negocio. Esto puede implicar definir métricas de evaluación, como la precisión de las predicciones o la capacidad de clasificar correctamente los diferentes tipos de compras de vehículos. El nivel de desempeño mínimo debe ser establecido por los participantes relevantes, como los responsables de marketing, gerentes de ventas o analistas de la industria automotriz.

Despliegue en producción: Para desplegar el modelo en producción, se requiere de:

a. Implementación de infraestructura: Es necesario contar con la infraestructura adecuada para alojar el modelo y realizar las predicciones en tiempo real. Esto puede incluir servidores, plataformas de nube, bases de datos y conectividad de red.

b. Integración de datos: Los datos necesarios para realizar las predicciones deben ser integrados al sistema de producción. Esto puede implicar la conexión con bases de datos de ventas, registros de clientes, características de los vehículos, entre otros.

c. Implementación del modelo: El modelo debe ser implementado en el entorno de producción, ya sea a través de código personalizado o mediante el uso de una plataforma específica para la implementación de modelos de aprendizaje automático.

d. Pruebas: Antes de lanzar el modelo en producción, se deben realizar pruebas para garantizar su funcionamiento adecuado. Esto puede incluir pruebas de integración, pruebas de rendimiento y validación cruzada con conjuntos de datos de prueba.

Procesos de monitoreo del desempeño en producción: Una vez que el modelo está en producción, es fundamental establecer procesos de monitoreo continuo para evaluar su desempeño y detectar posibles problemas o degradación del rendimiento. Algunos aspectos a considerar incluyen:

a. Supervisión de las predicciones: Se deben monitorear las predicciones realizadas por el modelo y compararlas con los resultados reales. Esto permite identificar discrepancias y evaluar la precisión y confiabilidad del modelo en el entorno de producción.

b. Actualización periódica: El modelo puede requerir actualizaciones periódicas para mantener su rendimiento óptimo. Esto puede implicar la incorporación de nuevos datos, reentrenamiento del modelo o ajustes de parámetros.

c. Seguimiento de métricas clave: Es importante definir métricas clave para evaluar el desempeño del modelo en producción, como la precisión, el error promedio o el nivel de confianza. Estas métricas deben ser monitoreadas de manera regular y comparadas con los niveles de desempeño establecidos anteriormente.

d. Retroalimentación de los usuarios: Es valioso recopilar retroalimentación de los usuarios o los equipos que utilizan el modelo en producción. Esta retroalimentación puede ayudar a identificar problemas o áreas de mejora, y permitir la actualización y optimización del modelo.

8. Conclusiones

1. Los modelos predictivos tanto supervisados como no supervisados son una alternativa de solución a un problema que se presenta en los concesionarios de automóviles al momento de realizar compras de autos.
2. El accuracy es una métrica de machine learning que sólo muestra una cara de la moneda en cuanto a modelos de clasificación. Es importante tener en cuenta otras métricas de desempeño como F1-Score.
3. Si bien asegurar que las muestras de la variable objetivo estén balanceadas puede ser un desafío importante para una compañía dada la naturaleza de los negocios, es importante tener en cuenta que cuando las clases están desbalanceadas, nuestros modelos predictivos van a tender a clasificar más aquella clase con la cual se tiene más muestras. Razón por la cual es determinantemente importante la obtención de más muestras de ambas clases.
4. Ante un problema de clasificación con clases desbalanceadas, es de vital importancia agregar hiperparámetros que penalicen dicho desbalance para evitar un overfitting.

Bibliografía

Faysal, Will Adams, Will Cukiersk. (2011). *Don't Get Kicked!* Kaggle. https://kaggle.com/competitions/DontGetKicked

(S/f). Epizy.com. Recuperado el 27 de mayo de 2023, de http://oramosp.epizy.com/teaching/202/topicos/clases/3_Regularizacion.pdf?i=