Opportunities at MeLi Code Excercise Data Scientist

Felipe Rojas Abatte

September 25, 2023

Introducción

En el contexto del Marketplace de MercadoLibre se necesita un algoritmo para predecir si un artículo listado en el mercado es **nuevo** o **usado.** Sus tareas implican el análisis de datos, el diseño, el procesamiento y el modelado de una solución de aprendizaje automático para predecir si un artículo es nuevo o usado y luego evaluar el modelo a partir de datos de prueba disponibles. Para ayudar en esa tarea, se proporciona un conjunto de datos en **MLA_100k_checked_v3.jsonlines** y una función **build_dataset** para leer ese conjunto de datos en **new_or_used.py**. Para la evaluación, utilizará la métrica de precisión para obtener un resultado de 0,86 como mínimo. Además, deberá elegir una métrica secundaria adecuada y también elaborar un argumento sobre por qué se eligió esa métrica.

Entregables

- El archivo, incluido todo el código necesario para definir y evaluar un modelo.
- Un documento con una explicación sobre los criterios aplicados para elegir las características, la métrica secundaria propuesta y el rendimiento alcanzado en esas métricas.
- Opcionalmente, puede entregar un análisis EDA con otro formato como .ipynb

Selección de features

El análisis del conjunto de datos provenientes del archivo **MLA_100k_checked_v3.jsonlines** originalmente contenidos en formato json, por lo cual, fue necesario realizar un preprocesamiento de datos considerando la transformación del archivo json en un dataframe, posterior **aplanamiento** de algunas columnas anidadas (non_mercado_pago_payment_methods, pictures) y finalizando con la limpieza de carácteres especiales como contenidas en otras columnas (sub_status, deal_ids, variations, attributes, tags, coverage_areas, descriptions, shipping.methods, shipping.tags). Este proceso fue realizado a través de la función `clean_flattern_json` del código python.
Una vez realizado este proceso se lograron reconocer 3 tipos de datos: datos numéricos, categóricos y booleanos. El primer filtro o criterio de eliminación de columnas del dataframe fue considerar aquellas que:

- tuvieran datos faltantes,
- tuvieran valores nulos o NaN
- tuvieran espacios en blanco
- tuvieran el mismo dato replicado en todas las filas (0 % de variabilidad)
- tuvieran en cada fila un dato distinto (100 % de variabilidad)

El segundo criterio de eliminación de columnas del dataframe fue considerar aquellas que aportaban exactamente la misma información, por ejemplo:

- seller_address.country.name contiene la misma información que seller_address.country.id
- seller_address.state.id contiene la misma informacion que seller_address.state.name
- seller_address.city.id contiene la misma informacion que seller_address.city.name
- non_mercado_pago_payment_methods.id contiene la misma informacion que .description

El tercer criterio de eliminación fue considerar la importancia de las variables utilizando a través del uso de un algoritmo de tipo random forest. Los resultados obtenidos muestran que las variables asociadas a fecha: dia, semana, mes y año además de diff_time son las que menos aportan al modelo, por lo tanto no las consideraremos en el entrenamiento.

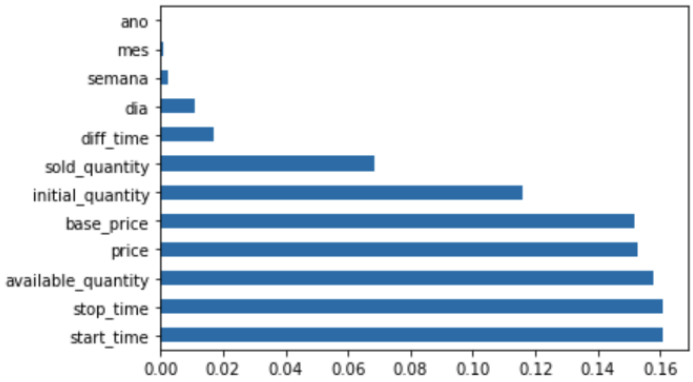

Figure 1: Importancia entre variables continuas

El cuarto criterio de eliminación fue realizar un análisis de correlación sobre las variables continuas para identificar aquellas que presentan mayor independencia, y de esa manera reducir el riesgo de multicolinealidad. El resultado muestra que las variables *base_price* y *price* presentan un alto grado de correlación. De la misma forma las variables *initial_quality* y *available_quality* también presentan un alto grado de

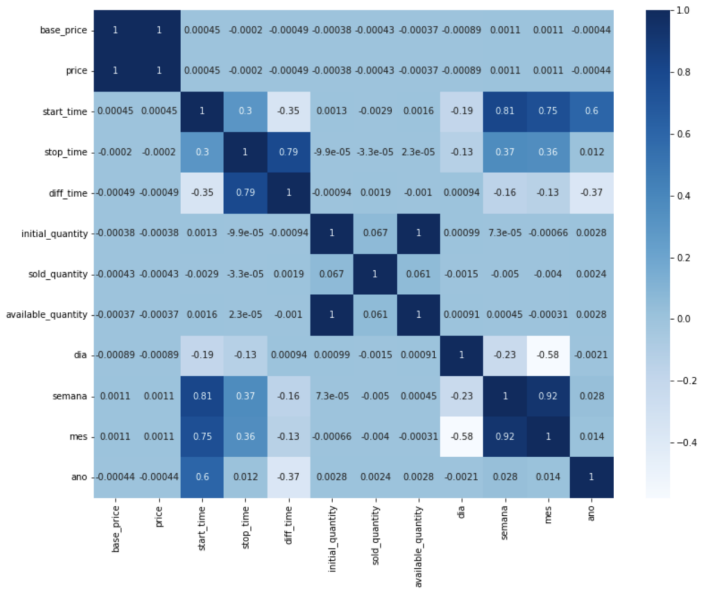

Figure 2: Correlación entre variables continuas

correlación.
Finalmente no consideraremos toda la sección de variables *pictures* dado a que contienen información con mucha variabilidad. De esta menera las variables a considerar para el entrenamiento del modelo son:
Numerical features

- stop_time
- start_time
- available_quantity
- price
- initial_quantity
- sold_quantity

Categorical features

- buying_mode
- accepts_mercadopago
- currency_id
- automatic_relist
- status
- seller_address.state.name
- seller_address.city.name
- shipping.local_pick_up
- shipping.free_shipping
- shipping.mode
- non_mercado_pago_payment_methods.description
- no_mercado_pago_payment_methods.type

Segunda métrica de desempeño

La segunda métrica de evaluación para el desempeño del modelo que utilizaré será **classification_report**. Esta métrica muestra un resumen completo del desempeño de un clasificador utilizando en su conjunto varias métricas de evaluación. Ayuda a evaluar la capacidad del modelo para clasificar correctamente diferentes clases, proporcionando información sobre el rendimiento de cada clase y la calidad general del modelo. La métrica de **classification_report** tiene varias ventajas por sobre la **Accuracy**:

- Proporciona una evaluación más detallada del rendimiento del modelo. Considera dentro del reporte métricas como precisión, recall y F1-score para cada clase, lo que brinda más información sobre el rendimiento del modelo en clases individuales. La Accuracy, por otro lado, solo proporciona un valor único que representa el rendimiento general del modelo.
- Manejo de datos desequilibrado. Cuando se trabaja con conjuntos de datos desequilibrados, donde algunas clases tienen menos datos que otras, la Accuracy puede ser engañosa. Un modelo puede funcionar bien en la clase mayoritaria pero mal en las clases minoritarias y aun así tener una alta precisión. El informe de clasificación proporciona métricas para cada clase, lo que facilita la identificación y solución de este problema.
- Equilibra la precisión y recall: la puntuación F1 en el informe de clasificación proporciona un equilibrio entre precisión y recall, dando igual importancia a los falsos positivos y los falsos negativos. Esto es particularmente importante en situaciones donde ambos tipos de errores tienen consecuencias similares.
- Identifica áreas de mejora. El informe de clasificación ayuda a identificar clases en las que el rendimiento del modelo es deficiente, proporcionando información sobre posibles áreas de mejora. Al centrarse en las métricas de cada clase, se pueden comprender mejor las debilidades del modelo y realizar los ajustes necesarios.

Al aplicar Ambas métricas de evaluación del desempeño del modelo Random Forest los resultados se muestran a continuación

Random Forest

```
rf_model = RandomForestClassifier(n_estimators=300, random_state=0)
rf_model.fit(X_train, y_train)
y1_pred = rf_model.predict(X_test)

print(confusion_matrix(y_test, y1_pred))
print('\nReporte de Clasificacion:')
print(classification_report(y_test, y1_pred))
accuracy1 = 100*accuracy_score(y_test, y1_pred)
print('Accuracy: {:.2f}%'.format(accuracy1))
```

```
[[4678  728]
 [ 659 3935]]

Reporte de Clasificacion:
              precision    recall  f1-score   support

         new       0.88      0.87      0.87      5406
        used       0.84      0.86      0.85      4594

    accuracy                           0.86     10000
   macro avg       0.86      0.86      0.86     10000
weighted avg       0.86      0.86      0.86     10000

Accuracy: 86.13%
```

Figure 3: Metricas de desempeño del modelo seleccionado (Random Forest)