Visión Artificial

Introducción

Equipo del proyecto: Iryna Pukas, Leonardo Zarza, Natasha Rocha, Percy Campos.
Objetivos:

- realizar un estudio amplio de los datos, definir los nulos y elegir la manera de trabajar con ellos, encontrar errores dentro de los datos de la empresa, encontrar patrones de errores y errores que vienen con los datos;
- identificar áreas de mejora y aplicar soluciones para mejorar la precisión y fiabilidad de la información;
- crear una manera óptima para almacenar los datos de la empresa a través de base de datos SQL y definir un modelo relacional que va a servir para calcular los KPI y hacer consultas sobre los datos;
- calcular las métricas de negocio propuestas en el guion e investigarlos (a través de las herramientas de visualización en particular) para crear consejos que pueden ayudar a la empresa;
- probar varios modelos de machine learning sobre los datos, principalmente analizando las imágenes, considerando un modelo aceptable cuando alcanza más de 80% de accuracy;
- investigar diferentes maneras de construir una red neuronal al final llegando a una que va a ser capaz de acertar en más de 90% de los datos.

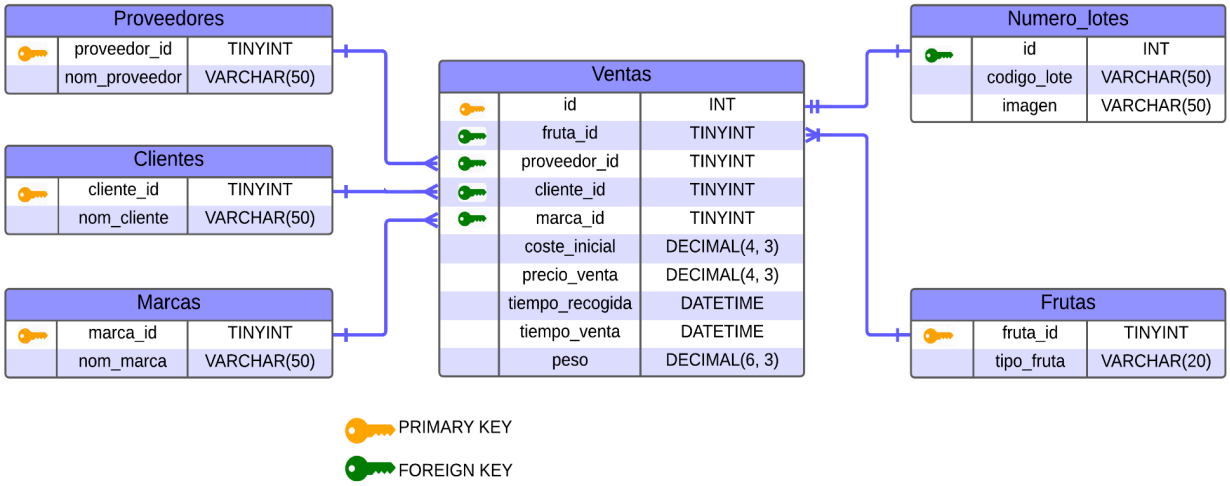
Las tareas desempeñadas por cada integrante:
Iryna: EDA, estructura de base de datos, KPI en python y SQL, machine learning, deep learning, visualización en Tableau, presentación.
Leonardo: la columna de categoría a través de las carpetas, investigar sobre los números de lotes y modelo de machine learning que define el tipo de fruta por lote, presentación.
Natasha: KPI, visualizaciones de Tableau, presentación.
Percy: estructura de base de datos, modelo relacional, KPI, presentación.

1. Modelo relacional

El propósito de la base de datos SQL es almacenar los datos sobre los lotes vendidos por la empresa Pontia Logista. La tabla principal, **ventas**, incluye la información principal de cada lote vendido por la empresa en el mes de septiembre. En columnas cliente_id, precio_venta, y tiempo_venta usamos DEFAULT NULL es por qué existe el concepto de la fruta dañada (no se ha podido vender), esos lotes no tienen precio de venta, tiempo de venta y cliente.

También optamos por elegir VARCHAR en varios casos, ya que se adapta a la longitud de diferentes variables (como fruta, marca, proveedor, cliente, que pueden variar bastante en el número de caracteres, de 10 hasta 45). Elegimos guardar 3 números decimales en variables

numéricas, por qué al importar los archivos al Workbench y guardar solo 2 decimales, la elección más típica, algunos valores se redondean (por ejemplo, valor de coste inicial 1,99 se redondea a 2), lo que hace variar ligeramente las métricas de negocio que sacamos también en python.



Las tablas de frutas, marcas, proveedores y clientes se utilizan para no repetir información en la tabla principal, ventas. Así que de esta manera optimizamos el rendimiento de las queries, ya que guardamos los nombres en respectivas tablas y en la tabla de ventas nos referimos al número que corresponde al nombre particular. La tabla numero_lotes sirve para consultar información específica sobre los lotes, cuál en caso de las queries no necesitamos.

2. Limpieza de datos

Abarcamos el proceso de la exploración de los datos con entender la estructura de los archivos JSON y pasarlos a dataframes de pandas (apartado 2 de EDA). El **primer error** que encontramos es la repetición de algunos títulos de las imágenes que se sitúan en diferentes carpetas, ya que en total tenemos 15 diferentes tipos de fruta, 3 de los cuales contienen también diferentes subtipos. Decidimos recorrer las carpetas de las imágenes y guardar sus nombres en la columna “categoria” (apartado 3.1).

Al unir los dataframes encontramos un **error**. En algunos casos el tipo de la fruta es “apple” mientras la imagen se sitúa en la carpeta de “guava A”. Al revisar algunas imágenes y comprobar el error, cambiamos el tipo de esos lotes a “guava” (apartado 3.3). Procedemos a investigar métricas comunes y en el proceso encontramos otro error, la repetición de algunos datos en todas columnas menos el tipo de la fruta. Decidimos eliminar esos datos, ya que no sabemos cuáles de ellos son verdaderos (apartado 3.5.1).

Procedemos a investigar el dataframe, aplicando diferentes tipos de visualizaciones para comprender mejor la dinámica de los datos. Encontramos que **cada marca produce cada tipo de fruta**, lo que contradice una afirmación del guion (apartado 3.5.2). Investigamos otra

cuestión del guion sobre el peso de la fruta vendida a cada cliente cada día (apartado 3.5.3). Investigamos el concepto de la fruta dañada (apartado 3.6). Según el guion, los lotes que no tienen el tiempo de venta se consideran **fruta dañada**. El porcentaje de ese tipo de fruta es de 0.3% dentro de todos los datos.

Pasamos a la investigación de los **nulos** (apartado 3.7). Decidimos eliminar los datos donde los nulos de las columnas de coste inicial y precio de venta coinciden. En los demás casos decidimos rellenarlos con la media de la columna correspondiente para cada fruta en particular. Descubrimos que la media y la mediana de esas dos columnas no tienen variaciones significativas.

Investigamos la **diferencia entre tiempo de venta y tiempo de recogida**. Resulta que en algunos casos existe diferencia negativa entre esos dos valores, lo que es imposible. Por eso decidimos eliminar esos valores (apartados 3.8 y 3.10).

Procedemos a la investigación de los **outliers** de los datos numéricos. Nos encontramos con valores negativos en la columna de peso, los que decidimos eliminar (apartado 3.9.1). En el caso de los precios no encontramos valores negativos, aunque notamos cierta tendencia en la cantidad de outliers dentro de top-3 frutas (guayaba, manzana y kiwi). También investigamos outliers dentro de proveedores, marcas y clientes, aunque las diferencias entre diferentes representantes no destacan significativamente (apartados 3.9.2, 3.9.3 y 3.9.4).

Otro **error** encontramos a la hora de investigar los números de lotes, marcas y frutas (apartado 3.11). En este caso vemos el error de dos tipos de fruta (Apple D, Apple E) de tener el mismo número de lote, lo que no es posible según el guion. Investigamos que la cantidad de los caracteres de los números de lotes varía. No encontramos ese error dentro de la columna de marcas.

Al final, calculamos los **KPIs** en Python y en SQL y obtuvimos los mismos resultados. También proponemos algunos de nuestros KPIs que creemos que pueden ser valiosos para la empresa.

3. Metodología de ML

Al principio, definimos que vamos a realizar la **clasificación** de fruta utilizando las imágenes, por eso utilizamos todas las imágenes a nuestra disposición. Recogemos las rutas, cargamos las imágenes y cambiamos su tamaño por dos razones: muchas imágenes tienen un tamaño ligeramente diferente y, teniendo el tamaño medio de 300x300, nos resulta muy costoso computacionalmente probar diferentes modelos. Al final, llegamos a la conclusión de utilizar el **tamaño de 50x50**. También aplicamos el análisis de componentes principales, **PCA**, para reducir la dimensionalidad de los datos. Codificamos las etiquetas de la fruta utilizando **Label Encoder**.

Utilizamos tres diferentes modelos de **Random Forest**: uno con datos sin utilizar PCA, este modelo alcanzó accuracy de 85% con 93 árboles, pero el tiempo de entrenamiento llegó a ser cerca de 5 horas; el segundo modelo se entrena en datos con utilización de PCA, su tiempo de

entrenamiento es de 30 minutos y alcanza accuracy de 77% con 99 árboles. Estos dos modelos nos dan la base para definir la cantidad óptima de valores, elegir los rangos de parámetros y optar por utilizar PCA. El tercer modelo de Random Forest se entrena sobre un total de 31.080 valores (2.072 valores por cada fruta, ya que decidimos equilibrar la cantidad de imágenes por cada fruta) y alcanza accuracy de 83% con 119 árboles sobre el conjunto de test y el tiempo de entrenamiento de 2,5 horas.

El siguiente modelo que probamos es **KNeighborsClassifier** con GridSearch sobre la cantidad de vecinos. Llega a accuracy de 85% con solo un número de vecinos (no profundizamos en ese modelo, pero no estamos seguros si es del todo correcto). Modelos **RidgeClassifier** y **SGDClassifier** alcanzan en accuracy 65% y 38%, los valores más bajos, así que fueron descartados. Al pensarlo, ambos modelos son lineales por naturaleza, y accuracy de entrenamiento nos muestra que no son adecuados para este caso cuando las relaciones entre las características y etiquetas no son linealmente separables.

Los mejores resultados muestran los modelos de **XGBoost**, 86%, y **LightGBM**, 87%. En estos modelos también aplicamos GridSearch para encontrar la combinación más óptima de variables (el parámetro de máxima profundidad y del número de estimadores). En matriz de confusión notamos que los valores que más se confunden por esos modelos son de los tipos de fruta que tienen varios subtipos (por ejemplo, manzana que tiene 5 subtipos y todos de ellos son diferentes según las imágenes).

Al final implementamos un modelo **LightGBM** sobre 16.000 valores (664 por cada tipo y subtipo), apartado LightGBM de machine learning. Ese modelo alcanzó la mayor accuracy, 88%, permitiéndonos mejorar el reconocimiento de diferentes subtipos.

También investigamos **los números de lotes** y descubrimos ciertos patrones dentro de esos códigos alfanuméricos. Utilizamos esos conocimientos para construir un modelo de Random Forest que por los 5 primeros caracteres y proveedor es capaz de predecir el tipo de fruta con accuracy de 95%.

En la parte de deep learning probamos varios modelos y varias combinaciones de datos. Combinamos los datos aleatoriamente, sin controlar el número de datos de cata tipo de fruta, y también controlando las cantidades de datos. Comprobamos, otra vez, que con regularización de la cantidad de datos de tipos de fruta obtenemos mejores resultados. También en comparación con la resolución de las imágenes que aplicamos en los modelos anteriores pudimos aumentar la resolución hasta 100x100, ya que al comprobar llegamos a la conclusión que es un valor óptimo que obtiene buenos resultados en entrenamiento y test y no tarda un período de tiempo exagerado para entrenarse.

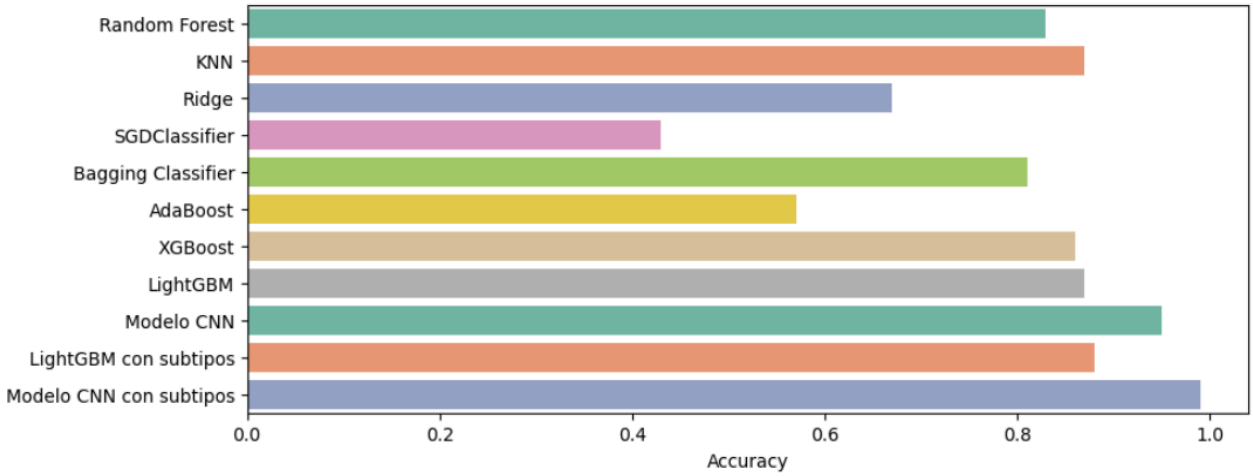
Al principio optamos por basarnos en capas convolucionales para el reconocimiento de imágenes. El **primer modelo** (0.1) consiste de 3 capas convolucionales, 1 capa densa y la capa de salida y llega a 91% de accuracy.

El **modelo 1** (el notebook deep_learning) es el que ha mostrado los mejores resultados sin aplicar definición de subtipos. Muestra accuracy de 95% sobre los datos de test. Ya que nos

quedan bastantes datos fuera de train/test (no están equilibrados según el tipo de fruta), al testearlos obtenemos accuracy de 93%. Aunque en la matriz de confusión notamos que, por ejemplo, la manzana se confunde bastante con otros tipos de fruta. Deducimos que la razón de ello es que la manzana en particular tiene 5 subtipos que son muy diferentes. Pensamos que la solución está en definir también subtipos (en manzana, guayaba y kiwi).

Por esa razón desarrollamos **modelo 6** (y a partir de ello diferentes variaciones de ese modelo para poder explorar más). Modelo 6 define no solo los tipos de fruta, pero también los subtipos. En este modelo alcanzamos la mayor accuracy en test, 99%, en 40 épocas. En la matriz de confusión podemos observar que los tipos de fruta que más se confunden son plátano con pera, manzana F con tomate, kiwi A con pera. Aunque al observar las imágenes en los que se ha confundido el modelo, podemos decir que esas imágenes en sí pueden resultar un poco confusas.

4. Comparación de modelos



Como se puede observar, modelo CNN con subtipos tiene la exactitud más alta, lo que sugiere que es un modelo de rendimiento excepcionalmente bueno. Seguido por modelo CNN (el que excluye definición de subtipos) con el porcentaje de acierto ligeramente más bajo. En general podemos concluir que a la hora de calcificación de imágenes, las redes neuronales convolucionales son la mejor elección.

Entre otros modelos, LightGBM y XGBoost alcanzan un porcentaje de accuracy bastante alto, de 86% y 87%. Al probar LightGBM con subtipos conseguimos aumentar accuracy hasta 88%. Esto es el máximo que podemos alcanzar en modelos de machine learning.

Modelos como Random Forest, Bagging Classifier y KNN Classifier alcanzan la accuracy ligeramente más baja, un poco por encima de 80%, lo que hace estos modelos aceptables según nuestra propuesta original. Otros modelos no llegan a superar 80% de accuracy que definimos como requisito para considerar un modelo satisfactorio.