Sofía Challenge

Francisco Sanhueza Matamala, para postular al cargo de Data Engineer

21/05/2024

Todo el código está subido en el repo

https://github.com/Pan-con-queso/sofia-challenge

Contexto

- Tenemos 25000 datos de usuarios que están siguiendo el protocolo de ejercicios y nutrición Blueprint de Brian Johnson y queremos hacer la regresión de que valores son mas influyentes en el vo2_max
- El vo2_max es un buen predictor de longevidad. Indica el volumen de oxigeno consumido al hacer un minuto ejercicio intenso. Se mide en ml/min/kg (Mililitros por minuto dividido en la masa corporal)

Brian Johnson, creador del programa Blueprint

El Dataset

- Datos de suplementos y cantidad de ejercicio hecho (98 columnas)
- Edad, Género y Nombre

Consideraciones Generales

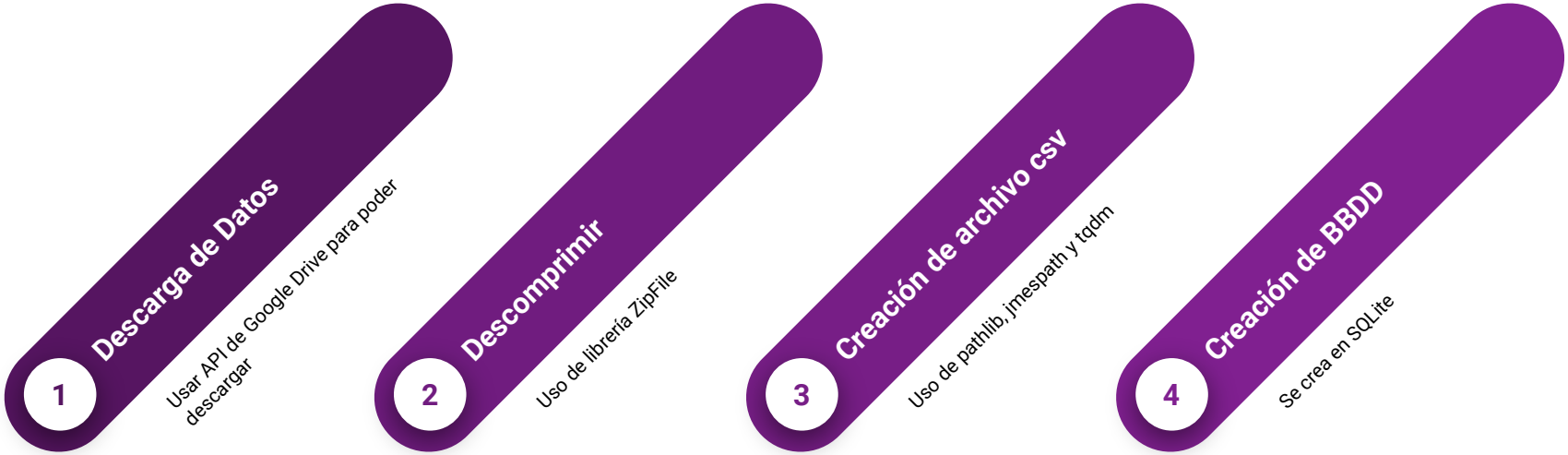
- Todo se programó en Python
- Se ejecutan todos los códigos en un computador Lenovo Thinkpad T480 año 2018 con Windows 10.
- Se optan por soluciones sencillas
- Los códigos están documentados y haremos referencia a ellos a lo largo de la presentación.

Estructura de la solución

1. Data Engineering
2. Predictive Modelling
3. Data Visualization
4. Bonus

1. Data Engineering

Partes de la solución



Descarga de Datos

- Tenemos que habilitar la API de Google Drive
- Autorizar consentimiento de credenciales oAuth 2.0 para aplicación en Google.
- Agregar correo personal como usuario test de la aplicación que va a usar la API de google drive
- Crear cliente oAuth 2.0
- Descargar archivo de credenciales
- Modificar quickstart en python https://github.com/googleworkspace/python-samples/blob/main/drive/quickstart/quickstart.py

gdrive_downloader.py

Descomprimir datos

- Se usa la librería zipfile
- El código es una aplicación directa de la librería
- Demora 9 minutos

`zipfile` — Work with ZIP archives ¶

Source code: Lib/zipfile/

The ZIP file format is a common archive and compression standard. This module provides tools to create, read, write, append, and list a ZIP file. Any advanced use of this module will require an understanding of the format, as defined in PKZIP Application Note.

This module does not currently handle multi-disk ZIP files. It can handle ZIP files that use the ZIP64 extensions (that is ZIP files that are more than 4 GiB in size). It supports decryption of encrypted files in ZIP archives, but it currently cannot create an encrypted file. Decryption is extremely slow as it is implemented in native Python rather than C.

The module defines the following items:

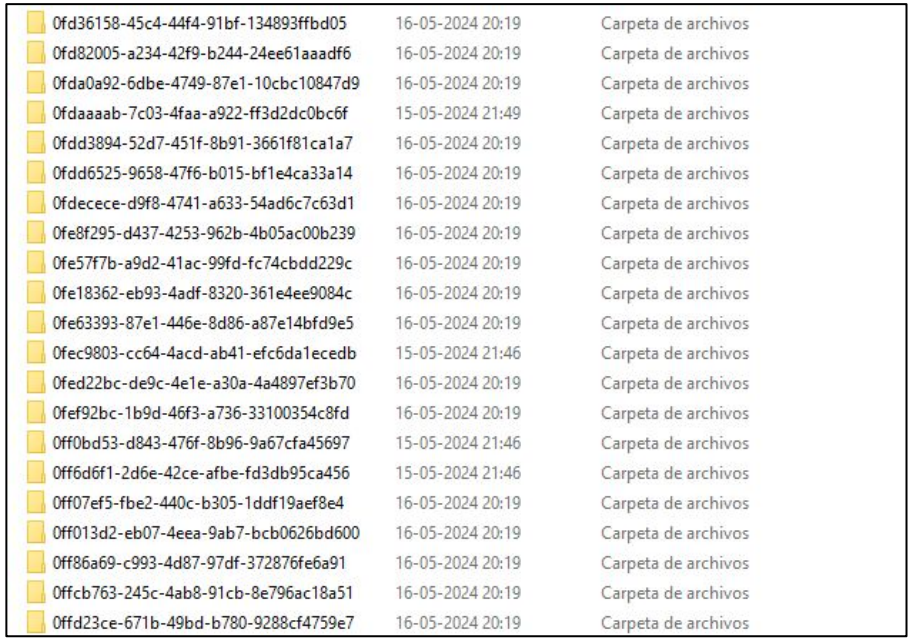
zip_extractor.py

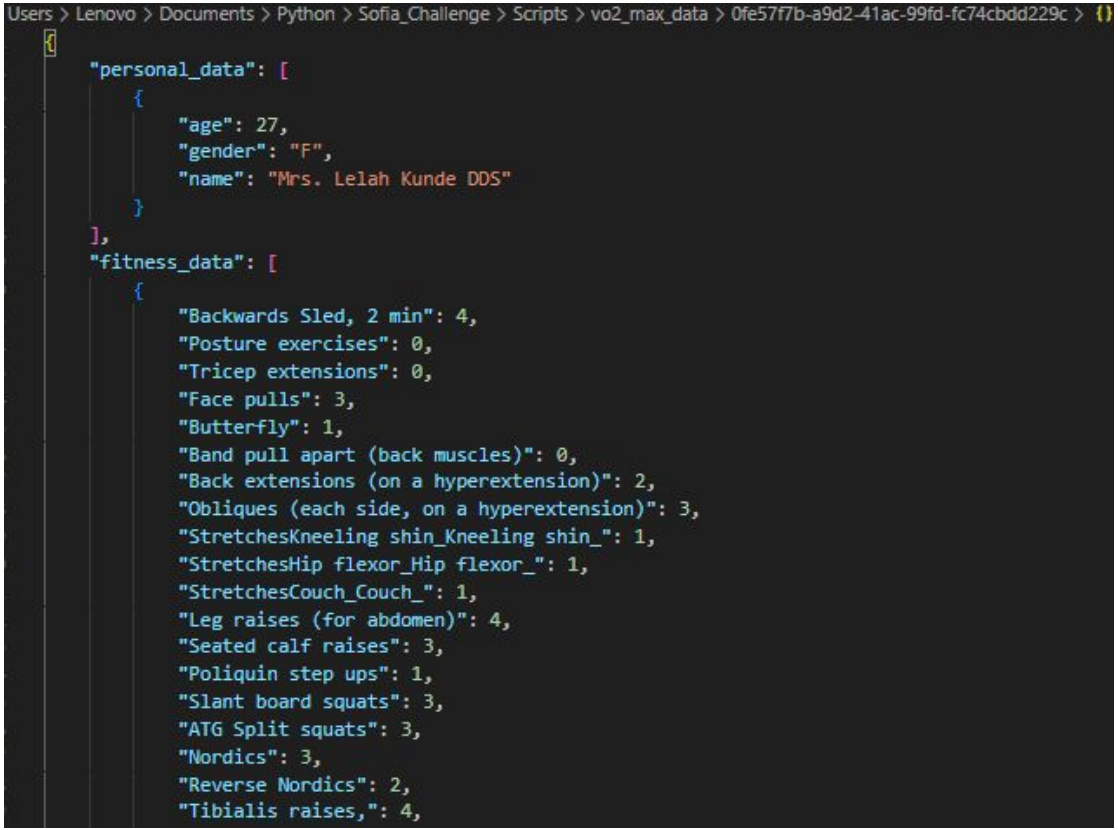
Escritura en csv

La estructura del zip descomprimido es la siguiente:

- uuid de usuario 1
  - archivo json con el nombre del uuid
- uuid del usuario 2
  - archivo json con el nombre del uuid

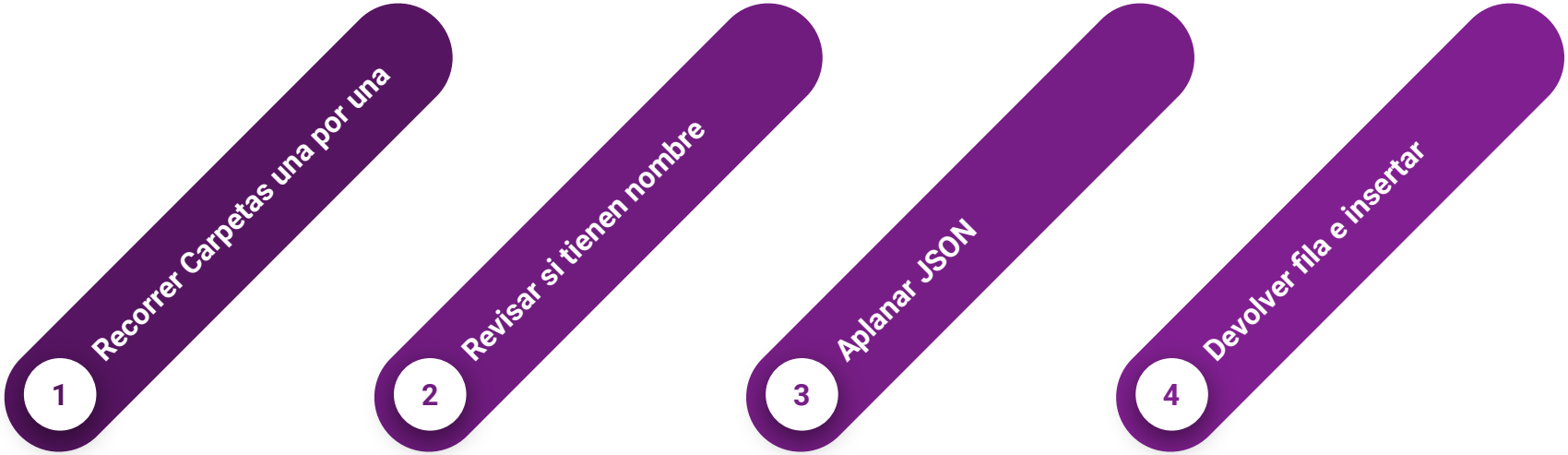
- etc





Y los JSON se veían así, como podemos ver, son diccionarios anidados

Partes de la solución



build_dataset.py

Inserción en SQLite

- Se usa cliente de SQLite para python .
- Se procesa la misma fila que nos entrega el recorrido por carpetas, pero se cambian todos los caracteres especiales por guiones bajos y se cambian los nombres de las columnas que empiezan con número.
- Se crea tabla solo si no existe
  - `f"CREATE TABLE IF NOT EXISTS vo2_max({row_string})"` Donde Row_string es la fila procesada como un string.
- Para insertar datos `"INSERT INTO vo2_max VALUES ({row_string})"`

build_dataset.py

Últimas consideraciones

- Cada vez que se revisa un archivo, se crea una copia vacía con extensión .DONE en caso de que tengamos que repetir la inserción desde algún punto
- Demoró 28 minutos la creación de todo el dataset en csv, pero se intertan exactamente los 25000 datos.
- Existen datos con nombres repetidos.

Mejoras

- Orquestar los 3 scripts ocupando herramientas como Airflow o un script que llame a los 3 scripts secuencialmente
- Definir esquema de SQLite de forma explícita, definir llaves primarias y hacer chequeos sobre los datos.
- Paralelizar iteración por las carpetas o probar técnicas de batch inserting.
- Arreglar inserción de archivo csv pues si se crea desde cero cada vez que se corre el script.

2. Predictive Modelling

- Todo el codigo de esta parte está en el jupyter notebook predictive_modelling.ipynb

Explorar datos

- Se exploran datos usando df.explore y df.types
- No hay datos faltantes ni columnas con tipos de datos inconsistentes, es un muy buen dataset!

Preprocesamiento y columnas nuevas

- Se hace one-hot-enconding para la columna de género
- Se elimina la columna de nombre, no tiene valor predictivo
- Se agrega columna is_elderly para personas mayores de 60 años

Modelos entrenados

- Se prueba una regresión lineal y un arbol de decisión.
- No se escalan los datos para la regresión lineal debido a que están en escalas distintas.
- Se prueba un XGBoost Regressor pero no se prueba más porque la ejecución toma varios minutos, y se privilegian modelos rápidos de revisar
- Se prueban las regresiones con el dataset completo, puesto que el interés **para esta parte de la presentación** es ver que variables son más importantes.

Subconjuntos de datos entrenados

- Solo Hombres
- Sólo Mujeres
- Solo mayores de 60 años

Se prueba con ambos modelos

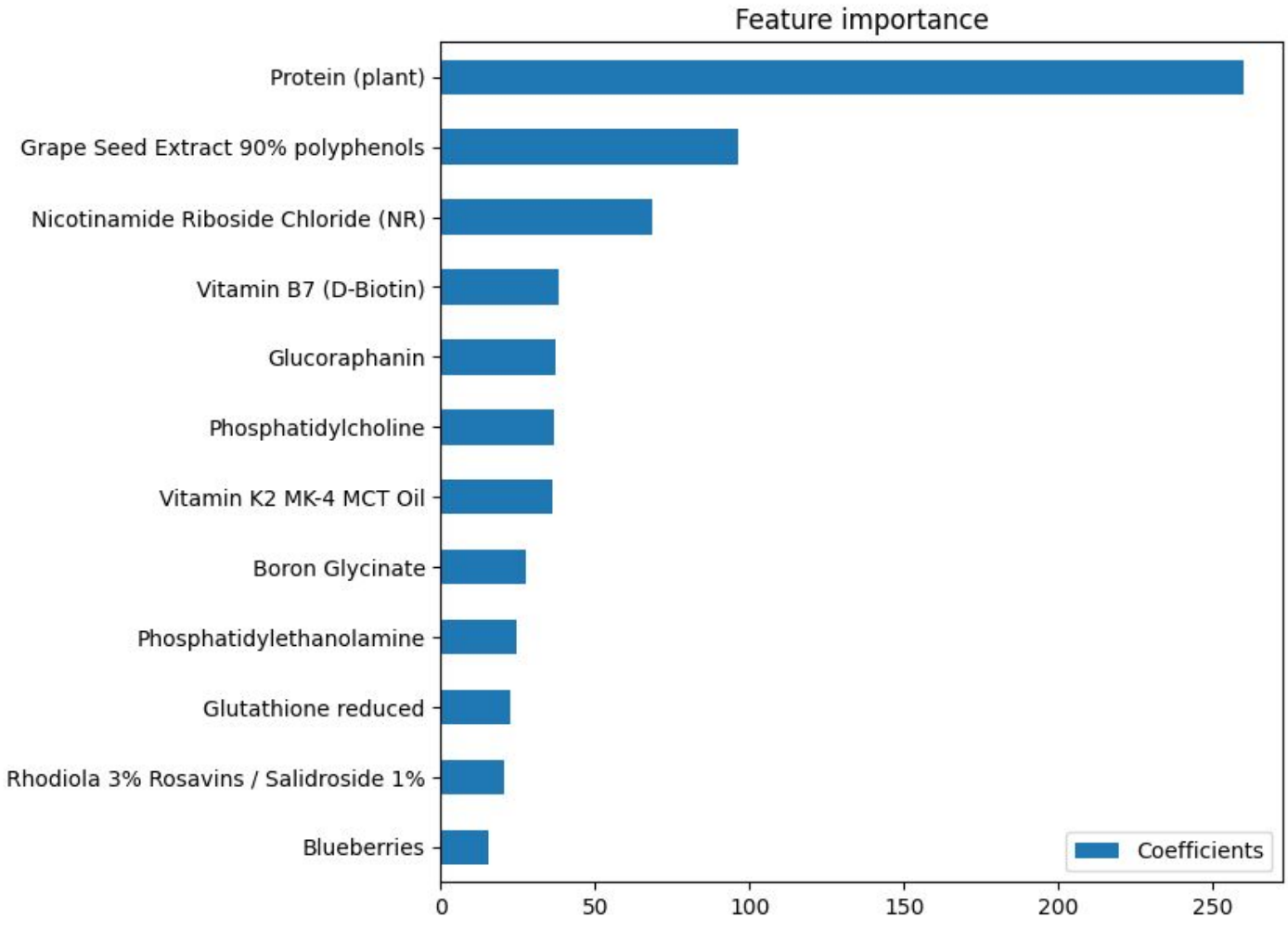
Resultados generales

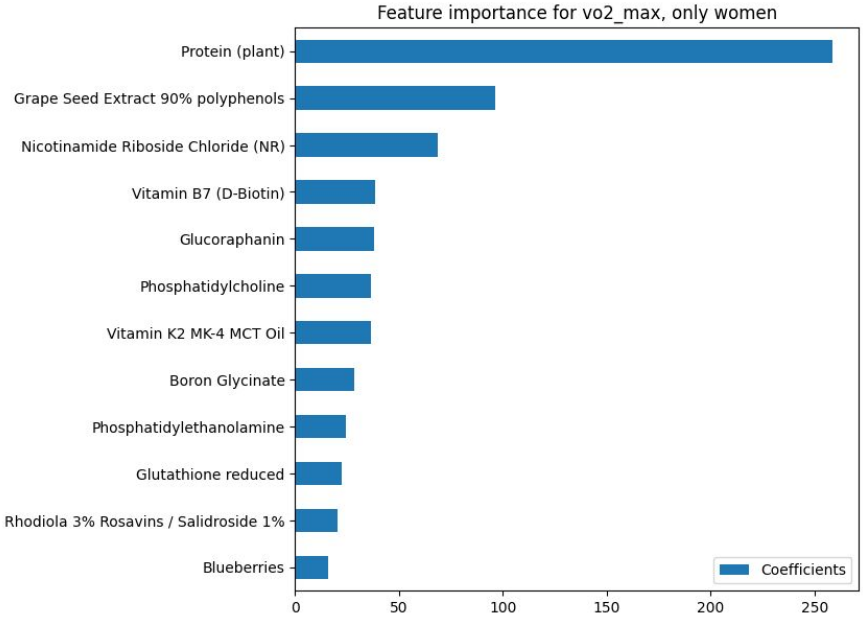
- Tanto para la regresión lineal como para el arbol de decisión, las variables mas importantes corresponden a algunos suplementos y la edad
- Para el árbol de decisión la edad es muy determinante en el vo2_max
- Para mayores de 60 años la edad empieza a ser ligeramente menos importante y aumentan más las influencias de ciertos suplementos alimenticios.
- No es notoria la diferencia entre la importancia de los atributos para hombres y mujeres
- En la regresión lineal no aparece la edad como variable más importante, pero puede ser por la escala de los datos.

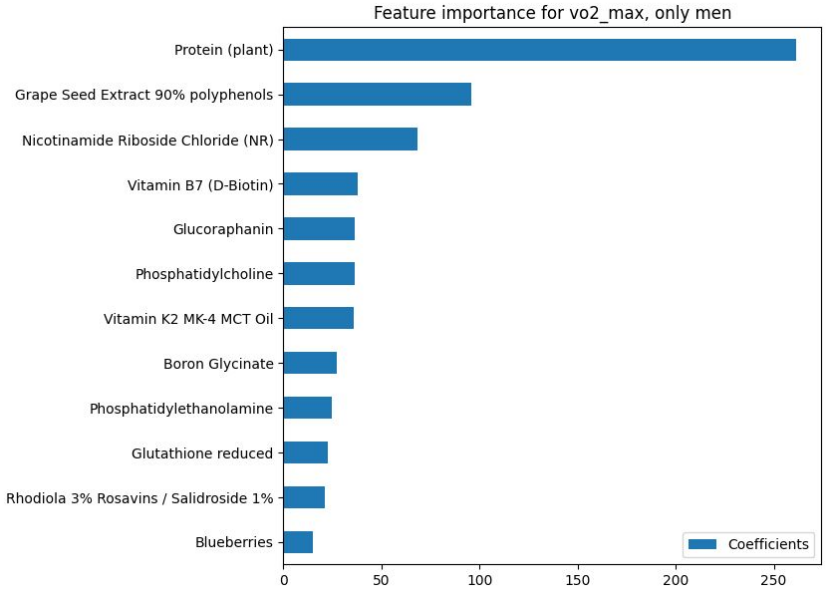
3. Data visualization

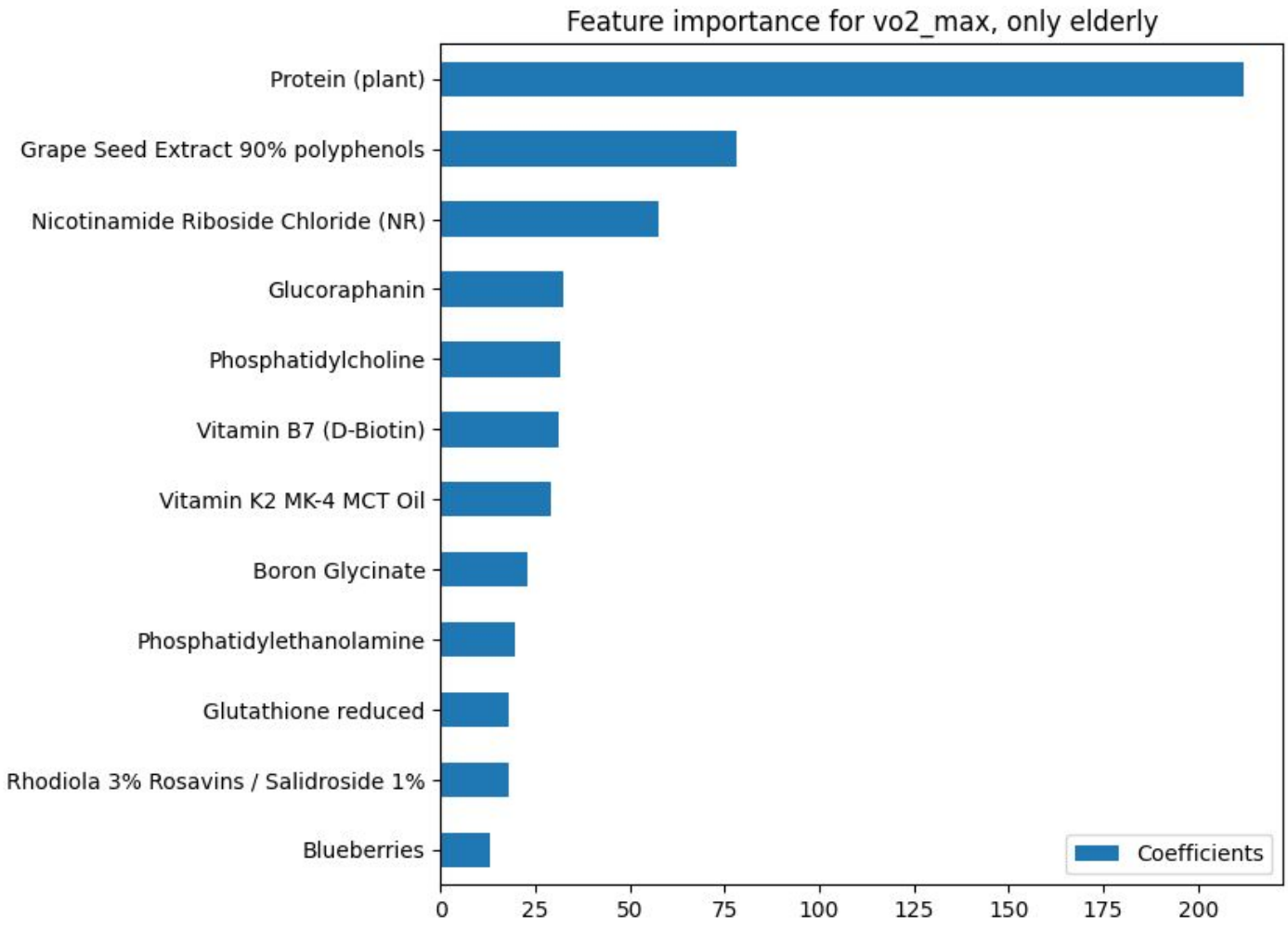
- Todo el codigo de esta parte está en el jupyter notebook predictive_modelling.ipynb

Regresión lineal

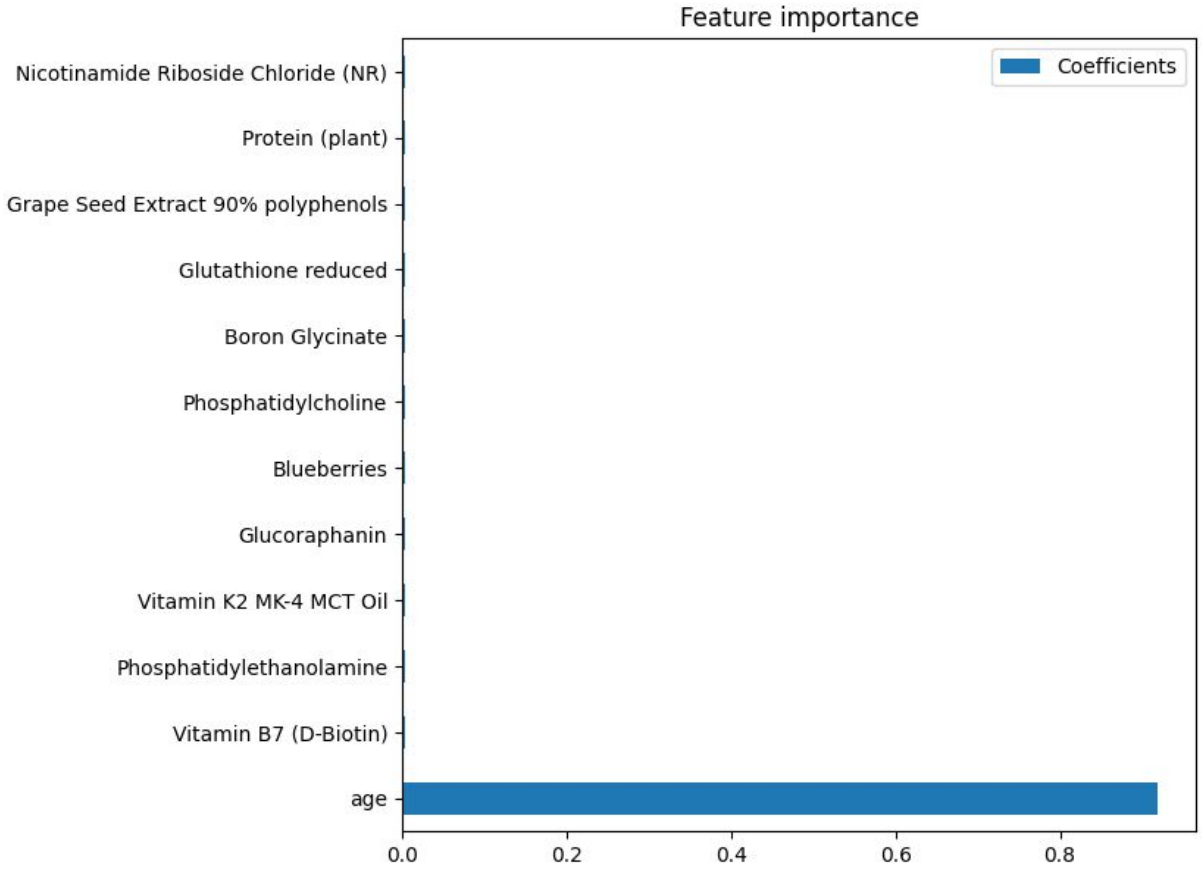

Feature importance
Protein (plant)
Grape Seed Extract 90% polyphenols
Nicotinamide Riboside Chloride (NR)
Vitamin B7 (D-Biotin)
Glucoraphanin
Phosphatidylcholine
Vitamin K2 MK-4 MCT Oil
Boron Glycinate
Phosphatidylethanolamine
Glutathione reduced
Rhodiola 3% Rosavins / Salidroside 1%
Blueberries
Coefficients
0
50
100
150
200
250


Feature importance for vo2_max, only women
Protein (plant)
Grape Seed Extract 90% polyphenols
Nicotinamide Riboside Chloride (NR)
Vitamin B7 (D-Biotin)
Glucoraphanin
Phosphatidylcholine
Vitamin K2 MK-4 MCT Oil
Boron Glycinate
Phosphatidylethanolamine
Glutathione reduced
Rhodiola 3% Rosavins / Salidroside 1%
Blueberries
Coefficients
0
50
100
150
200
250

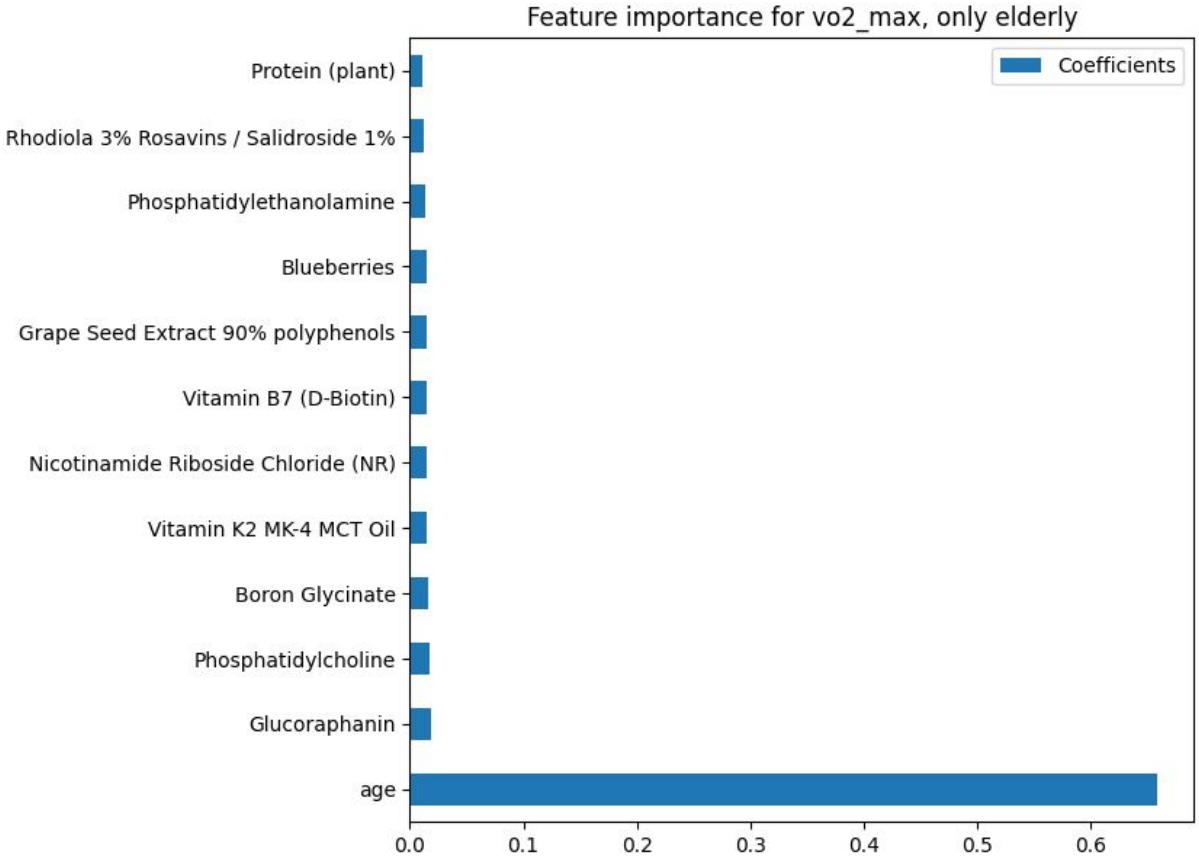

Feature importance for vo2_max, only men
Protein (plant)
Grape Seed Extract 90% polyphenols
Nicotinamide Riboside Chloride (NR)
Vitamin B7 (D-Biotin)
Glucoraphanin
Phosphatidylcholine
Vitamin K2 MK-4 MCT Oil
Boron Glycinate
Phosphatidylethanolamine
Glutathione reduced
Rhodiola 3% Rosavins / Salidroside 1%
Blueberries
Coefficients
0
50
100
150
200
250


Feature importance for vo2_max, only elderly
Protein (plant)
Grape Seed Extract 90% polyphenols
Nicotinamide Riboside Chloride (NR)
Glucoraphanin
Phosphatidylcholine
Vitamin B7 (D-Biotin)
Vitamin K2 MK-4 MCT Oil
Boron Glycinate
Phosphatidylethanolamine
Glutathione reduced
Rhodiola 3% Rosavins / Salidroside 1%
Blueberries
Coefficients
0
25
50
75
100
125
150
175
200

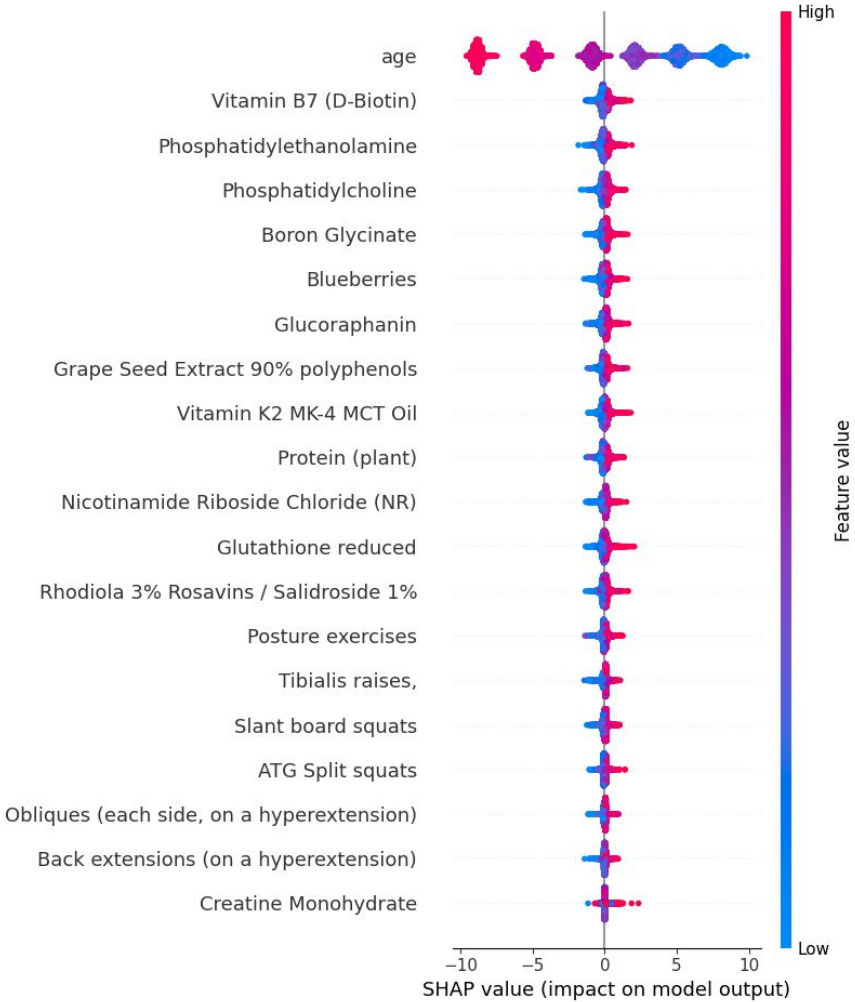
Decision Trees



Variables más importantes para Arbol de Decisión, dataset completo



Variables más importantes para Arbol de Decisión, sólo mayores



Intenté sacar la importancia de los features del modelo de regresión lineal usando Shapley Values, y me entregó un resultado muy parecido, la edad es altamente determinante en el resultado final, tanto negativamente como positivamente y las otras variables son las de suplementos alimenticios.

No calculé importancia de conjuntos de features.

4. Bonus

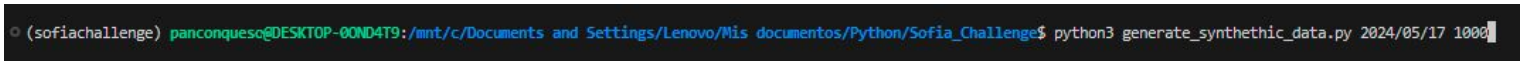
- Todo el codigo de esta parte está en la carpeta bonus/

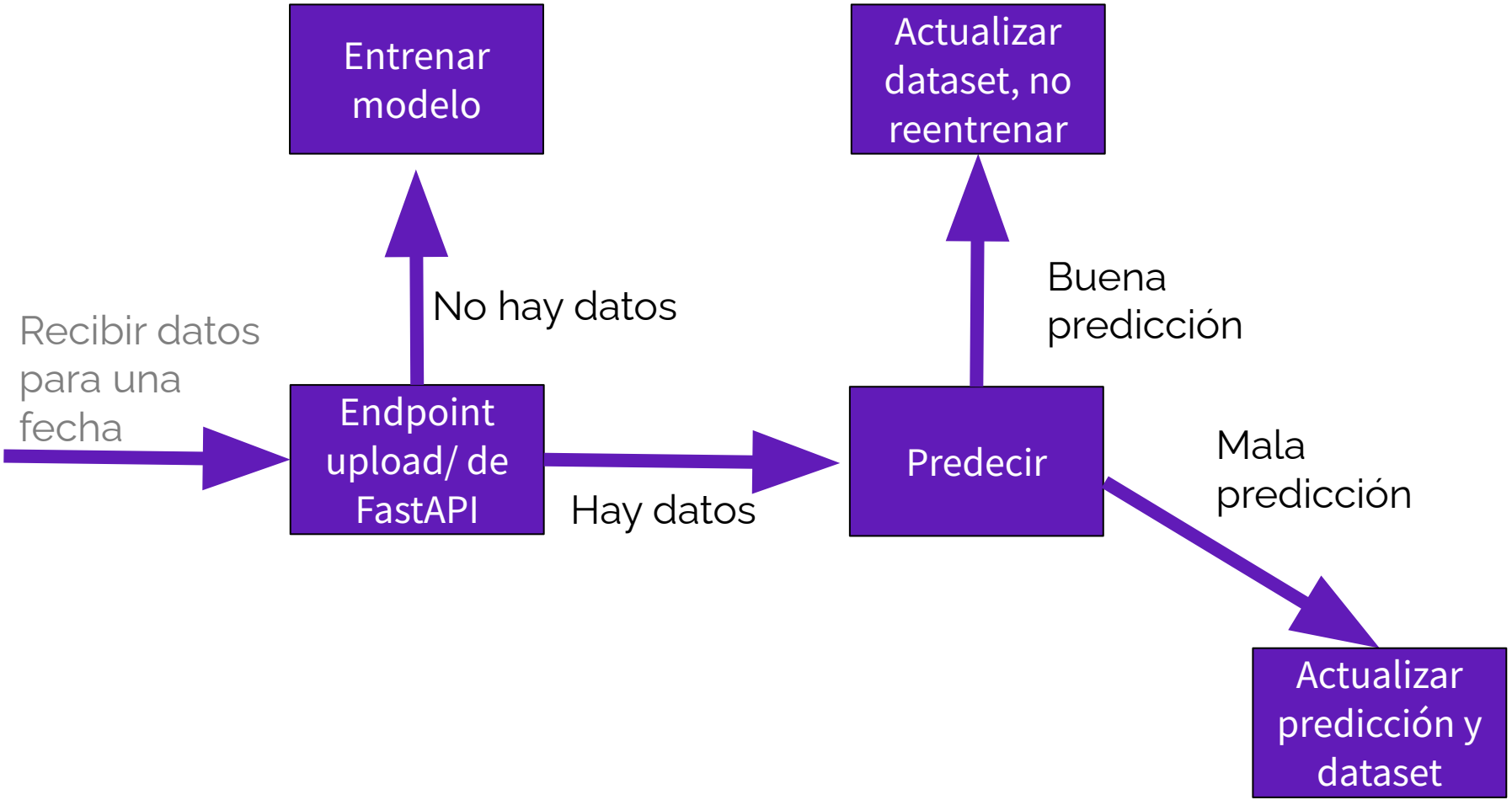
Síntesis

- Se implementa de inicio a fin una app de FastAPI y una clase Model que puede recibir un modelo pre entrenado.
- Se reescribe gran parte de lo que hice en el Jupyter Notebook como métodos de la clase Model.
- Se prueba el modelo dividiendo los datos completos en 25 archivos de 1000 filas porque simular los datos nos daban cosas poco realistas.

Generador de datos sintéticos

- Programa de Python que recibe una fecha en string y el número de datos.
- Todos los valores son random entre 0 y el optimal dose del spreadsheet de excel linkeado.




Entrenar modelo
Actualizar dataset, no reentrenar
No hay datos
Buena predicción
Recibir datos para una fecha
Endpoint upload/ de FastAPI
Hay datos
Predecir
Mala predicción
Actualizar predicción y dataset

¿Cómo actualizar la predicción?

- Actualizar cada vez que baje el score

```
INFO:     127.0.0.1:46196 - "POST /update HTTP/1.1" 200 OK
0.9628190003817867
Updating dataset for future training
INFO:     127.0.0.1:46198 - "POST /update HTTP/1.1" 200 OK
0.9691671700746417
INFO:     127.0.0.1:46200 - "POST /update HTTP/1.1" 200 OK
0.9651416722960895
Updating dataset for future training
INFO:     127.0.0.1:46202 - "POST /update HTTP/1.1" 200 OK
0.9696559798564769
INFO:     127.0.0.1:46204 - "POST /update HTTP/1.1" 200 OK
0.9621519274270373
Updating dataset for future training
```

Y por si fuera poco

Se crea un Dockerfile que está en el repo.

Con ese se puede subir a alguna nube de preferencia para poner el modelo en producción!

Posibles mejoras

- Probar otros modelos
- Predicciones con ventanas de tiempo
- Tipado fuerte del modelo con Pydantic.

Demo

¿Preguntas?