Blog post draft (English)

Group members (Add linkedin handle)

- Adrián Aguado García https://www.linkedin.com/in/adrianaguado/
- Arturo <to add LinkedIn or preferred SM handle>
- Mateo <to add LinkedIn or preferred SM handle>
- Olumayowa Onabanjo https://www.linkedin.com/in/olumayowa-onabanjo-452b8143
- Raquel Martínez Martínez https://www.linkedin.com/in/rakelmarmar/

Introduction

Cancer has life changing implications and brain metastases are currently one of the hardest to treat (see glossary for more). But there is hope, early detection and effective treatment go a long way to ensure patients have the best odds against the disease.

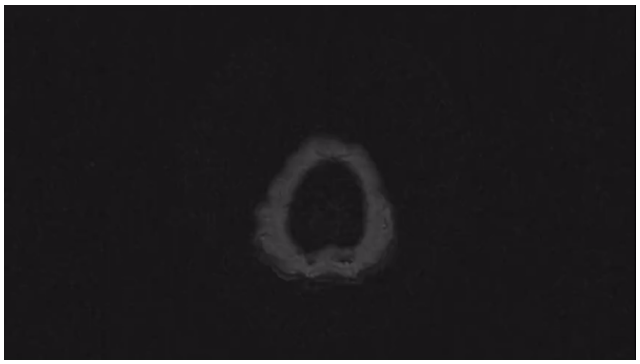
Objectives

One of our goals was to accurately predict how much time is left to savor life. The second goal is just as important, determining when a person is likely to relapse.

We'll run you through our dataset, how we explored the data, decided on an appropriate Machine Learning (ML) approach and our results. We hope this summary helps anyone interested in this dataset and we look forward to your feedback and comments.

Dataset

We worked on the MOLAB Brain Metastasis Dataset, a structured and anonymized dataset covering segmented Magnetic Resonance Imaging (MRI) scans, clinical data and morphological measurements for 75 patients.



Our dataset is made up of 3 groups of files with structured and unstructured data.

- *Clinical data* is an excel spreadsheet with patient data and treatment history.

- *Morphological measures* is also an excel sheet that contains dimensions of identified tumors from the MRI scan. One of the major benefits of working with this dataset was the opportunity to stand on the shoulders of the giants who segmented the MRIs and annotated them.
- Finally, the *MRI scans* themselves are contained in a zip file??? in various formats such as nifti <please review this last part>.

Clinical data

Variable	What it really means
Patient Data	This section includes data such as age at 1st scan and sex
Lesion	Tumors are numbered in the order they are discovered
Primary Tumor	Primary tumors are classified by number (1 - 12) and subtype for the most common classes (breast & lung cancer)
Whole Brain Radiation Therapy (WBRT)	As its name suggests (includes dosage and period administered)
Stereotactic Radiosurgery (SRS)	A non-surgical radiation therapy.
Radiation Necrosis	Tissue breakdown due to radiation
Surgery	Describes the type (full, partial, unknown) and time it was carried out
Systemic Treatment	Drugs given and the time period
Death	Time like for other variables is measured in days and the cause
MRI Follow up dates	Minimum 4 appointments, Maximum 19

Morphological measures

Variable	What it really means
General data	Patient ID number, information about the image and the equipment used to capture it.

Other measurement data (Don't worry too much about this. We aren't MRI experts either.)	• SBS (Space between Slices) in millimeters • SCS (Slice Thickness) in millimeters • REPTIME (Repetition Time) in milliseconds • CEVOLUME (Contrast-Enhancing Volume) VCE • NECVOLUME (Necrotic (Non-Enhancing) Volume) Vn • TOTALVOLUME V=VCE+VN • CERIMWIDTH (Contrast-Enhancing (CE) Spherical Rim Width): it's a function of the volume calculated from the average width of areas CE • SURFACEREGULARITY: the closer to 1 it gets the more spherical the tumor is. • MAXDIAMETER3D: maximum longitudinal measurement of the tumor

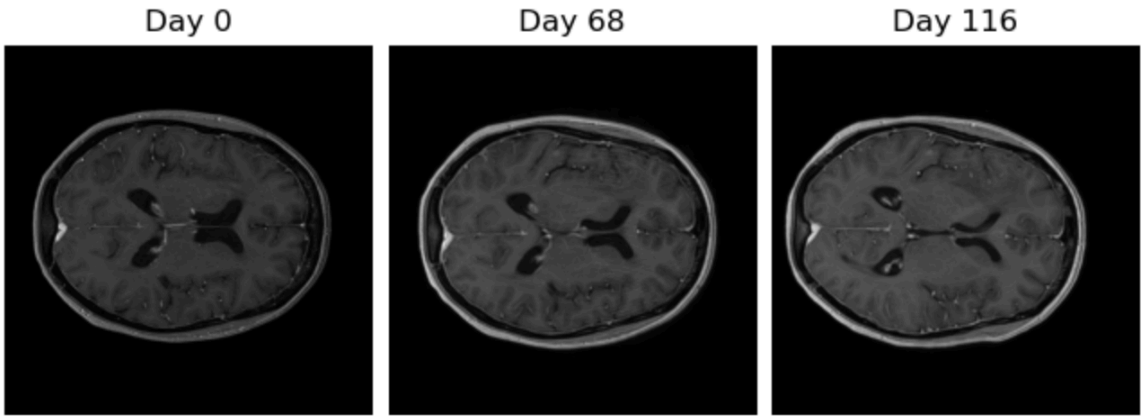
MRI scans

Since we were nowhere near having enough data to train our own GPT with images and text, we stuck to the structured part of our dataset <phew>.

If you're really bent on viewing the images, here's how <also add link to notebook>.

FUNCIÓN REPRESENTAR MRI

```python
import nibabel as nib

############################################## FUNCIÓN QUE MUESTRA LA IMAGEN DE UN MRI
def plot_mri_image(nii_file):
    # Cargar la imagen .nii
    nii_img = nib.load(nii_file)
    # Obtener los datos de la imagen
    nii_data = nii_img.get_fdata()

    # Calcular el punto medio del cerebro para cortes axiales
    mid_slice = nii_data.shape[2] // 2

    # Plotear las imágenes
    fig, axes = plt.subplots(1, 3, figsize=(15, 15))

    # Cortes Axiales
    axes[0].imshow(nii_data[:, :, mid_slice], cmap='gray')
    axes[0].axis('off')
    axes[0].set_title("Axial")

    # Cortes Sagitales
    sagittal_slice = nii_data.shape[0] // 2
    axes[1].imshow(nii_data[sagittal_slice, :, :].T, cmap='gray', origin='lower')
    axes[1].axis('off')
    axes[1].set_title("Sagittal")

    # Cortes Coronal
    coronal_slice = nii_data.shape[1] // 2
    axes[2].imshow(nii_data[:, coronal_slice, :].T, cmap='gray', origin='lower')
    axes[2].axis('off')
    axes[2].set_title("Coronal")

    plt.show()
```



Problem description

While working with this dataset we were conscious of some factors that could affect the accuracy of our model, such as, size, class imbalance, high dimensionality & outliers.

Size

Although our dataset is packed full of useful information, it's relatively small compared to other medical studies. This was however not a deterrent as prior studies show that quality data can be used to train machine learning algorithms with good results. It's also encouraging for those working on breakthroughs in rare diseases with small populations.

Class imbalance

Our data was skewed in terms of age, sex, primary tumor and cause of death (no surprises on the last one). Having a small dataset creates a compounded problem if you want to resample data to have an equal number of samples for each class.

We worked around this problem by…

High dimensionality

This was mainly introduced by the wide range of interventions applied over the course of treatment. Throwing every single variable you can at a ML algorithm is not the best approach given our dataset size. So, it's important to know which variables had the most influence on the outcome.

There are many great ways to find this, a Principal Component Analysis (PCA) is a good example. We used a…

Outliers

Given the size of our dataset, outliers (even just one) have a huge effect. Data visualization (graphs, plots, charts) made them easier to spot.

Exploratory data analysis (EDA)

You should never start training ML algorithms without a proper EDA, you'll encounter fewer errors and go over previous steps less often. This is how we got to know our data (steps will depend on your own dataset).

Pre-processing

Install/import libraries: It does sound obvious if you're not new to ML with Python, but you should download all your libraries from the beginning. Of course, you should know which libraries you'll need <add laugh smiley>. For that, we highly recommend you join the next Saturdays AI session near you <wink>.

Read files: Once again stating the obvious, but your method will depend on your environment and where your data is stored (shared cloud file, local file, etc.). Since we worked on Google Colab, our data was also stored on a shared drive. Our morphological measurements example does not have any null rows (this is an exception for ML datasets <smiley>).

```
df = pd.read_excel(r'/content/drive/MyDrive/OpenBTAI_MORPHOLOGICAL_MEASUREMEN
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 499 entries, 0 to 498
Data columns (total 18 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   PATIENT            499 non-null    int64
 1   TIME POINT         499 non-null    int64
 2   PIXELSPACING       499 non-null    float64
 3   SBS                499 non-null    float64
 4   STH                499 non-null    float64
 5   STATION NAME       499 non-null    object
 6   REPTIME            499 non-null    float64
 7   ECHOTIME           499 non-null    float64
 8   MAGNETIC FIELD     499 non-null    float64
 9   LESION             499 non-null    int64
 10  TOTALVOLUME        499 non-null    float64
 11  CEVOLUME           499 non-null    float64
 12  NECVOLUME          499 non-null    float64
 13  CERIMWIDTH         499 non-null    float64
 14  MAXDIAMETER3D      499 non-null    float64
 15  TOTALSURFACE       499 non-null    float64
 16  RENDVOLUME         499 non-null    float64
 17  SURFACEREGULARITY  499 non-null    float64
dtypes: float64(14), int64(3), object(1)
memory usage: 70.3+ KB
```

Data cleaning: Exactly as expected, there was still some data cleaning to be done to make the data worthy of ML training.

Firstly, the descriptions in the first few rows of the Clinical Data excel file and the merged cells in the column headings affected how the dataframe column names appeared. So that had to be fixed.

```
[ ] df1 = pd.read_excel(r'/content/drive/MyDrive/OpenBTAI_METS_ClinicalData_Nov26
    df1 = df1.drop(index=[160,161,162,163])
    df1.head()
```

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	EGFR
0	010005	40.0	1.0	1.0	1.0	1.0	1.0	NaN
1	010005	40.0	1.0	1.0	3.0	1.0	1.0	NaN
2	010020	44.0	1.0	NaN	1.0	1.0	4.0	NaN
3	010020	44.0	1.0	NaN	2.0	1.0	4.0	NaN
4	010020	44.0	1.0	NaN	3.0	1.0	4.0	NaN

UNNAMEDS:

- 0 = ID
- 1 = AGE AT MR1
- 2 = SEX, 1= FEMALE,2 = MALE
- 3 = GPA
- 4 = LESION
- 5 = TYPE
- 6 = SUBTYPE
- 44 = SURGERY TYPE
- 45 = SURGERY TIME
- 70 = COMMENTS
- 71 = DEATH OCURRENCE (1 = YES,0=NO)
- 72 = DEATH TIME
- 73 = DEATH CAUSE (1 BRAIN PROGRESSION, 2 SYSTEMIC ILLNESS, 3 = 1+2, 4 OTHER)

This was solved with a simple df.rename function.

```
df1 = df1.rename(columns={'Unnamed: 0':'ID','Unnamed: 1':'AGE','Unnamed: 2':
df1.head()
```

Completely null columns were the next to go. The code to check for null columns is shown below (which was generously shared by <Sofia's LinkedIn>).

FUNCIÓN COMPROBAR COLUMNAS NULAS

```
[ ]  # Comprobar columnas nulas
     def columnas_nulas(df):
             mis_val = df.isnull().sum()
             mis_val_percent = 100 * df.isnull().sum() / len(df)
             mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
             mis_val_table_ren_columns = mis_val_table.rename(
             columns = {0 : 'Valores faltantes', 1 : '% del Total de Valores'})
             mis_val_table_ren_columns = mis_val_table_ren_columns[
                 mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
             '% del Total de Valores', ascending=False).round(1)
             print ("El dataframe seleccionado tiene " + str(df.shape[1]) + " columnas.\n"
                 "Hay " + str(mis_val_table_ren_columns.shape[0]) +
                   " columnas que tienen Valores nulos.")
             return mis_val_table_ren_columns
```

```
[ ]  columnas_nulas(df1)
```

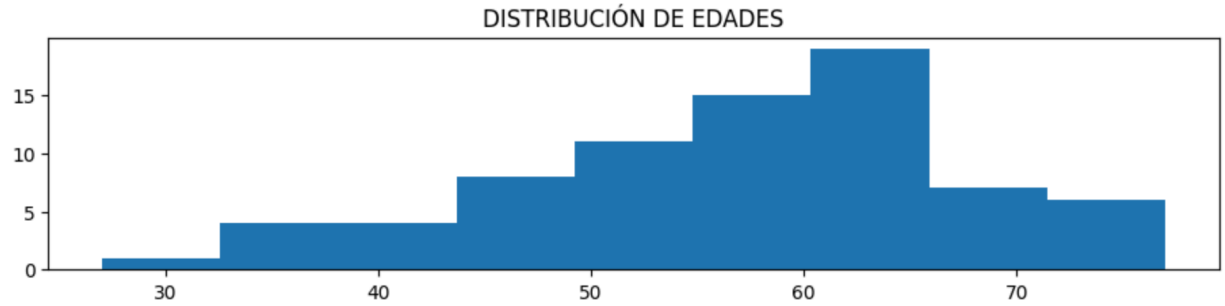
```
El dataframe seleccionado tiene 93 columnas.
Hay 71 columnas que tienen Valores nulos.
```

	Valores faltantes	% del Total de Valores
HER2	160	100.0
TTF1	160	100.0
Sinaptofisina	160	100.0
Dosis/Frac.2	159	99.4
Nfrac.4	159	99.4
PDL1	159	99.4
RE	158	98.8
COMMENTS	158	98.8
KRAS	158	98.8
RP	158	98.8
CKIT	157	98.1
BRCA1	156	97.5
Drug.7	156	97.5
End.12	156	97.5

Once we confirmed the columns that did not add value to the analysis, we used the df.drop function to remove these columns. This method preserves more useful data than heading

straight to df.dropna without checking which would take out rows/columns with some useful data.

Data characteristics: Although this can be a little more complicated for categorical data, it is good to know whether or not your data is skewed towards a particular class (for instance, more males than females, more people under or over 40, etc.). This is where the matplotlib library <link> comes in handy. <change image to show categorical data chart>



Data transformation to get new variables: To work with the scan dates, the original format was changed to datetime to get other variables such as days between scans. First the function to transform cells with string contents to datetime format is defined (datesdiff).

FUNCIÓN CALCULAR DIFERENCIA DE FECHAS EN DÍAS

```
def datesdiff(dfa,dfb):
  'FUNCIÓN QUE CALCULA LA DIFERENCIA DE FECHAS (EN DÍAS) ENTRE DOS ELEMENTOS DE UN DATAFRAME b-a'
  a = pd.to_datetime(dfa,format = "%Y%m%d") # Fecha 1
  b = pd.to_datetime(dfb,format = "%Y%m%d") # Fecha 2
  dif = (b-a).days
  return dif

  #EJEMPLO DE USO: datesdiff(df['TIME POINT'][0],df['TIME POINT'][1])
```

Then the function is called later to be executed on a numpy array. Remember creating functions is a really useful way to save lines of code for functions that will be reused.

```
a = np.array(df['TIME POINT']) # Array con las fechas
b = np.array(df['PATIENT'])    # Array con los pacientes

c = np.column_stack((a,b))

days0 = np.zeros_like(b) # Días entre sucesivos escáneres

days = np.zeros(75) # Días entre primer y último escáner
j = 0
k = 0
for i in range(1,len(c)):
  if c[i][1] == c[i-1][1]:
    days[j] += datesdiff(c[i-1][0],c[i][0])
    days0[k] = datesdiff(c[i-1][0],c[i][0])
  elif c[i][1] != c[i-1][1]:
    j += 1
  k += 1
```

In summary, the hype is true, data preprocessing does take a huge chunk of the ML training time. But it is worth it.

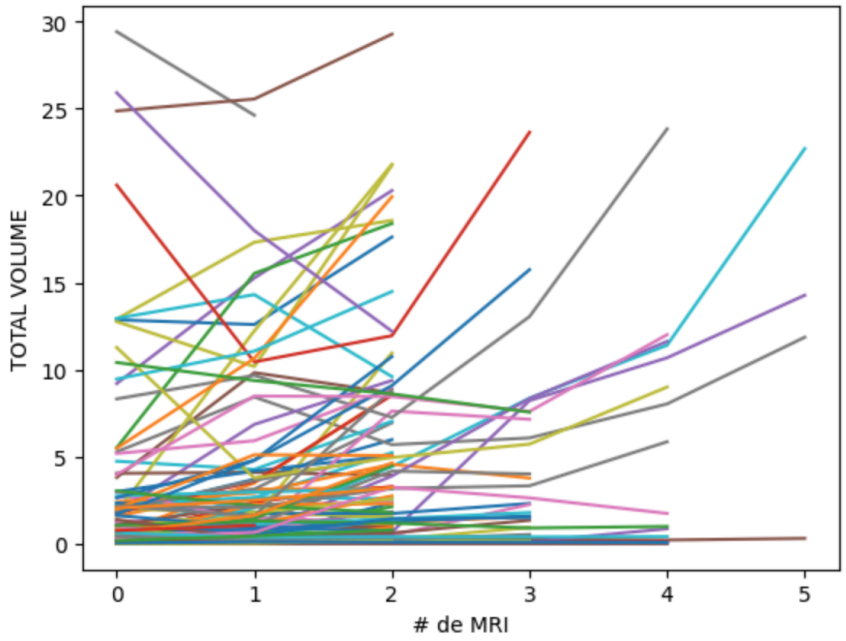
<working with libraries, preferences>

Data visualization

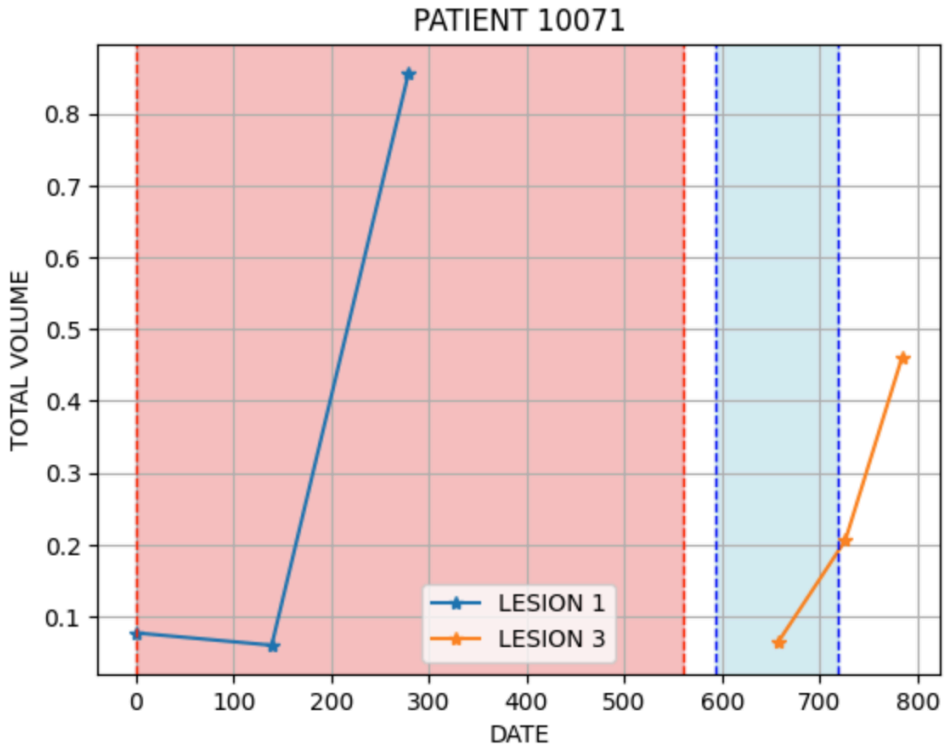
A lot of the prior work came together at this stage and informed our decisions for model selection and training. Also, it's easier to look at than lines of code. The correlation matrix for morphological measures is a good example. From this, we determined that TOTALVOLUME, CERIMWIDTH, and SURFACEREGULARITY were more significant variables to use for model training. We left out the other variables that are highly correlated because they are already summarized by another variable. For example, CEVOLUME is already contained in TOTALVOLUME.

	PATIENT	TIME POINT	LESION	TOTALVOLUME	CEVOLUME	NECVOLUME	CERIMWIDTH	MAXDIAMETER3D	TOTALSURFACE	RENDVOLUME	SURFACEREGULARITY
PATIENT	1.000000	0.053646	-0.171839	0.079648	0.041500	0.136389	-0.093757	0.086677	0.099292	0.079571	-0.236898
TIME POINT	0.053646	1.000000	0.082384	-0.009769	0.027001	-0.089293	0.063607	-0.032703	0.003397	-0.009718	-0.063226
LESION	-0.171839	0.082384	1.000000	-0.238300	-0.243727	-0.137701	-0.185144	-0.276982	-0.263953	-0.238313	0.139952
TOTALVOLUME	0.079648	-0.009769	-0.238300	1.000000	0.953802	0.733809	0.589308	0.873139	0.971274	0.999999	-0.377129
CEVOLUME	0.041500	0.027001	-0.243727	0.953802	1.000000	0.495805	0.729736	0.875902	0.951849	0.953724	-0.398272
NECVOLUME	0.136389	-0.089293	-0.137701	0.733809	0.495805	1.000000	0.053339	0.543258	0.655190	0.733981	-0.189537
CERIMWIDTH	-0.093757	0.063607	-0.185144	0.589308	0.729736	0.053339	1.000000	0.700798	0.652022	0.588997	-0.395243
MAXDIAMETER3D	0.086677	-0.032703	-0.276982	0.873139	0.875902	0.543258	0.700798	1.000000	0.948536	0.872836	-0.616076
TOTALSURFACE	0.099292	0.003397	-0.263953	0.971274	0.951849	0.655190	0.652022	0.948536	1.000000	0.971160	-0.507527
RENDVOLUME	0.079571	-0.009718	-0.238313	0.999999	0.953724	0.733981	0.588997	0.872836	0.971160	1.000000	-0.376828
SURFACEREGULARITY	-0.236898	-0.063226	0.139952	-0.377129	-0.398272	-0.189537	-0.395243	-0.616076	-0.507527	-0.376828	1.000000

We plotted the total volume of each tumor between MRI scans. While most of the tumors seem to be growing, there are periods of shrinking or slowed growth.



Looking closely at the events (treatments, surgery, or no interventions) during these periods, we hoped to reveal patterns. <annotate to highlight treatment periods>



Proposed approach

Taking a cue from our EDA, we added a new category for each tumor (its growth rate). That is, tumors that grew by more than 30% (considered a relapse) and those that didn't. Easy right? No, not really, we asked the experts. And we recommend you do too.

	PATIENT	LESION	TIME POINT	CERIMWIDTH	SURFACEREGULARITY	TOTALVOLUME	GPA	Sex	Age	RECAIDA
0	10005	1	19010618	0.338262	0.763884	0.162125	1.0	1.0	40.0	True
1	10005	1	19011011	0.446508	0.755049	0.372887	1.0	1.0	40.0	True
2	10005	3	19011011	0.084779	0.646738	1.604080	1.0	1.0	40.0	True
4	10005	3	19020128	0.132334	0.644168	3.134727	1.0	1.0	40.0	False
6	10020	1	19010822	0.368203	0.702189	0.359200	NaN	1.0	44.0	True

Model selection

Try everything, seriously, everything. If you don't have a lot of time and you have access to mentors who are good at training models then you should ask them.

Model training

Out of the 160 unique tumors, 154 had tumor dimensions available for training.

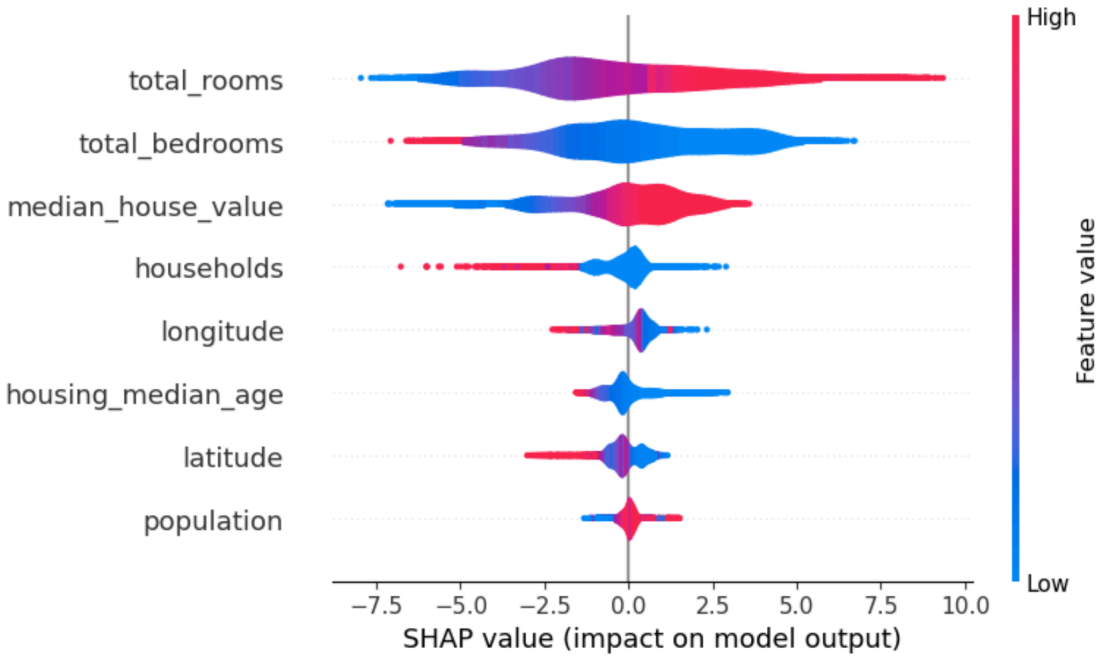
Results

<to update>

Explainable ML

Our model is designed to make predictions that are critical to people's lives and their decisions for the future. So we also made sure it would be transparent both to us and patients. To that end, we've added the SHAP (SHapely Additive exPlanations) values, links to the data and our code <link>. Why? Because we know it's harder to trust AI when it appears mysterious.

<this image is a placeholder, the plan is to have SHAP values for our own model>



Conclusion

Most of us knew little about Python and in just 12 weeks we trained our very own ML algorithm together. Hopefully, our story helps you start or continue your steps to making lives better with artificial intelligence.

We would like to thank our instructors, mentors, and the organizers of Saturdays AI Asturias for their help in guiding us on this journey.

Next steps

Next steps for anyone interested but that we didn't have time to explore

- Cause of death
- Varying % growth to define as a relapse

Tools

These are just a few of the tools we found useful while working on our project (apart from the ones mentioned earlier). Some are not strictly used for ML but are handy for conceptualization and information retrieval.

Glossary (Created with Bing Copilot)

Topic	Explanation
Cancer Cells	Some cells become abnormal—they keep growing and dividing uncontrollably.
Tumors	Abnormal cells group together to form lumps called tumors. Tumors can damage healthy tissues.
Metastasis	Cancer cells can break away and travel to other parts of the body, forming new tumors (metastasis).

Bibliography

- MOLAB brain metastasis dataset https://molab.es/datasets-brain-metastasis-1/?type=metasrd