Identifying Age-related conditions

Baseline Presentation

Akhila Ganti
Megan Nguyen
Leslie Nie
Kevin Stallone
Tim Tung

Berkeley
UNIVERSITY OF CALIFORNIA

Introduction

- The goal of our project is to predict if a person has any of the three age-related medical conditions. In order to assess, we will create a model trained on measurements of health characteristics.
- Aging is a risk factor for numerous diseases and complications. The growing field of bioinformatics includes research into interventions that can help slow and possibly reverse biological aging and prevent major age-related ailments.
- Data science has a role to play in developing new methods to solve problems with diverse data, even if the number of samples is relatively small.
- Data is sourced from an active (June 2023) Kaggle competition: https://www.kaggle.com/competitions/icr-identify-age-related-conditions/overview

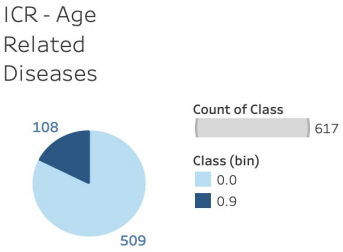
Berkeley
UNIVERSITY OF CALIFORNIA

- What/Why/How
- We will need to predict if the person has one or more of any of the three medical conditions (Class 1), or none of the three medical conditions (Class 0).
- EDA part 1 and part 2

EDA

Initial Observations:

- There is a training set of data, very limited test set and a meta data file.
- There are a total of 617 observations:
  a. 82.5% of class 0 (no age related disease) and
  b. 17.5% of class 1 (one of three age related condition).
- There seems to be an imbalanced target Class.
- There are 56 features in the data set for consideration.
- Small sample set of data - how best to split for training, validation and testing.
- Feature selection will be important as many variables don't seem to have much correlation with the target.
- There seems to possibly be some multicollinearity between features that we will be considering.
- Data cleanup will be needed to remove NaN values and unknown dates in the Epsilon field.



Berkeley
UNIVERSITY OF CALIFORNIA

- Synthetic Minority Oversampling Technique (SMOTE) is a statistical technique for increasing the number of cases in the dataset in a balanced way.
- Principal Component Analysis (PCA) is one of the most commonly used unsupervised machine learning algorithms across a variety of applications: EDA, dimensionality reduction, information compression, data de-noising and others.
- t-distributed stochastic neighbor embedding (t-SNE) is a statistical method for visualizing high-dimensional data by giving each datapoint a location in a two or three-dimensional map.tSNE
- Density plots for each feature in code to understand the features
- Imputer to impute values to replace NaNs

EDA – Features

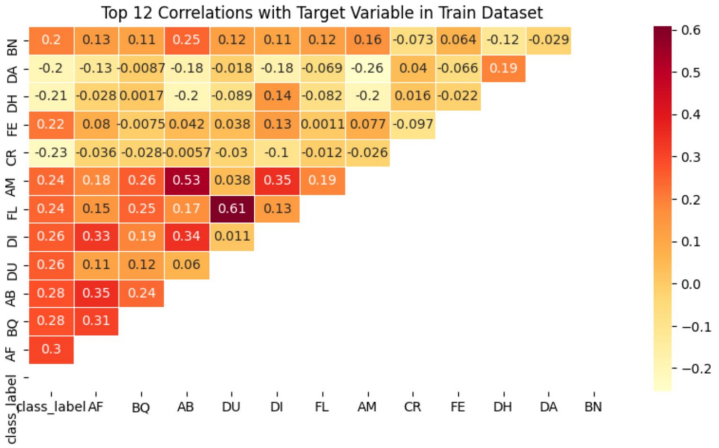
	data type	#missing	%missing	#unique	min	max	first quartile	second quartile	third quartile
Id	object	0	0.000000	617	NaN	NaN	NaN	NaN	NaN
AB	float64	0	0.000000	217	0.081187	6.161666	0.252107	0.354659	0.559763
AF	float64	0	0.000000	599	192.59328	28688.18766	2197.34548	3120.31896	4361.63739
AH	float64	0	0.000000	227	85.200147	1910.123198	85.200147	85.200147	113.73954
AM	float64	0	0.000000	605	3.177522	630.51823	12.270314	20.53311	39.139886
AR	float64	0	0.000000	130	8.138688	178.943634	8.138688	8.138688	8.138688
EH	float64	0	0.000000	127	0.003042	42.569748	0.003042	0.085176	0.237276
EJ	object	0	0.000000	2	NaN	NaN	NaN	NaN	NaN
EL	float64	60	9.724473	311	5.394675	109.125159	30.927468	71.949306	109.125159
EP	float64	0	0.000000	275	78.526968	1063.594578	78.526968	78.526968	112.766654
EU	float64	0	0.000000	455	3.828384	6501.26448	4.324656	22.641144	49.085352
FC	float64	1	0.162075	600	7.534128	3030.655824	25.815384	36.394008	56.714448
FD	float64	0	0.000000	337	0.29685	1578.654237	0.29685	1.870155	4.880214
FE	float64	0	0.000000	615	1563.136688	143224.6823	5164.66626	7345.143424	10647.95165
FI	float64	0	0.000000	498	3.58345	35.851039	8.523098	9.945452	11.516657
FL	float64	1	0.162075	388	0.173229	137.932739	0.173229	3.028141	6.238814
FR	float64	0	0.000000	435	0.49706	1244.22702	0.49706	1.131	1.51206
FS	float64	2	0.324149	161	0.06773	31.365763	0.06773	0.250601	0.535067
GB	float64	0	0.000000	560	4.102182	135.781294	14.036718	18.771436	25.608406
GE	float64	0	0.000000	264	72.611063	1497.351958	72.611063	72.611063	127.591671
GF	float64	0	0.000000	611	13.038894	143790.0712	2798.992584	7838.27361	19035.70924
GH	float64	0	0.000000	596	9.432735	81.210825	25.034888	30.608946	36.863947
GI	float64	0	0.000000	615	0.897628	191.194764	23.011684	41.007968	67.931664
GL	float64	1	0.162075	355	0.001129	21.978	0.124392	0.337827	21.978
class_label	int64	0	0.000000	2	0.0	1.0	0.0	0.0	0.0

- Anonymized Features
- Main features include:
  - class (if binary classification)
  - alpha (if multiclass classification)
  - AB-GL
- 16% of the features are missing values ranging from 0.1% to 10%
- All features are floats except for ID and EJ

Berkeley
UNIVERSITY OF CALIFORNIA

- As mentioned previously, this dataset comes from Kaggle's June 2023 Challenge
- All of the features in the dataset are anonymized. So we unfortunately don't have any insight into what the health characteristics are. For example, some of the features could be heart pressure, cognitive ability, or age as those would all play roles in to age related conditions, but we don't have that information
- The main features in our dataset are:
  - class which is a binary class of 0 or 1 depending on if the patient has been diagnosed with an age related condition.
  - Alpha which is a multiclass feature of four variables: A, B, C, and D which denote which age related condition someone has been diagnosed with. A means no diagnosis and the rest are different anonymized diagnoses
  - AB-GL which are our 56 anonymized health characteristics.
- Some of our features have missing values. Specifically 16% of them or 9 of the features. The biggest offenders are the features BQ and EL both of which are missing 60 values which is just under 10% of the total number of rows.
- All of our features are floats except for ID and EJ. ID is a unique identifier for each patient that consistents of numbers and letters and EJ is a binary class with either an A or a B for its value

EDA – Top Correlated Features



Lastly, here is a correlation heat map between our 12 highest correlated features which was created during our EDA process. From our initial findings, two of highest correlations I would like to point that we'll keep an eye on during our study are:

AM with AB

and FL with DU

Beyond that, we discovered that a majority of the health characteristics were not very correlated with each other.

Next Tim is going to talk about the how we will preprocess our data

Preprocessing

- Null Analysis:
  - Null values occur in nine of the features (primarily in features BQ [39] and EL [38]).
  - Given the relatively small dataset, we want to preserve as many observations as possible. Therefore, we will fill the null values rather than dropping observations.
  - Given that the features are all anonymized, we will use KNNImputer to fill the null values.
- Feature Reduction:
  - We will be using PCA/LDA and/or t-SNE to determine the most important features in the dataset and assist with dimensionality reduction.
  - We will also perform an ANOVA analysis and retain a sufficient number of features to explain at least 80% of the variance in the dataset.

Preprocessing

- Class Imbalance:
  - Given the significant class imbalance, we will attempt to use class weights to improve model performance.
  - We may also attempt to upsample the minority class by creating synthetic examples using SMOTE.
  - We will also use k-Fold Cross Validation with all of our models to utilize as much of the dataset as possible.
- Separation of train/validation/test Datasets:
  - We will use train_test_split() for dataset segmentation.
  - We will test both 70/30 splits and 80/20 splits to assess if there is a meaningful difference in model performance.
- Categorical Variables:
  - Feature “EJ” is the only non-numeric feature in the dataset (value A or B).
  - We will convert the feature to numeric values using 1-hot encoding.

Prediction Algorithms

- **Logistic Regression**: Models the relationship between predictor variables and the probability of a binary outcome using a logistic function. Suited for binary classification tasks, such as predicting medical conditions. We plan to use this as our baseline.
- **Decision Tree**: Built using Gini impurity as the splitting criterion. Shallow tree with a maximum depth of 3 to mitigate overfitting. Used for predicting age-related conditions.
    - **Bagging (Ensemble)**: Utilizes unpruned decision trees as base classifiers, with 500 trees trained on bootstrap samples. Final prediction is made by majority voting.
    - **AdaBoost (Ensemble)**: Employs stump decision trees as weak learners, with 500 decision stumps used. Weights are adjusted at each iteration to prioritize misclassified instances.
- **KNN**: Uses 5 nearest neighbors and Euclidean distance metric (Minkowski distance with p=2). Features are standardized with zero mean and unit variance using StandardScaler() before feeding into the model.
    - **Majority Vote (Ensemble)**: Combines predictions from a decision tree (max depth of 2, entropy criterion) and KNN models. Majority rule voting is performed using a user-defined MajorityVoteClassifier.
- **Neural Network**: Different configurations tested, including varying hidden layers, nodes, activation algorithms (tanh, relu), and optimizers (Adam, SGD). Weights are randomized during initialization and updated through backpropagation.

Berkeley
UNIVERSITY OF CALIFORNIA

- **Logistic Regression**: The core idea behind logistic regression is to model the relationship between the predictor variables and the probability of the binary outcome using a logistic or sigmoid function. The logistic function takes any real-valued input and maps it to a value between 0 and 1, which represents the probability of belonging to the positive class (e.g., presence of a medical condition) versus the negative class (e.g., absence of a medical condition). Medical conditions will be binary, such as the presence or absence of a medical condition or the occurrence of an event, so a Logistic regression is specifically designed for binary classification tasks, making it a natural choice for predicting medical conditions with two possible outcomes.This could be used as a baseline.
- **Decision Tree**: This is built using the Gini impurity as a criterion for making the splits and has a maximum depth of 3, meaning the tree is composed of decision nodes where the splits are made based on feature values, and leaf nodes which contain the prediction. This particular decision tree is relatively shallow (max depth of 3), which might make it less prone to overfitting. We intend to use this decision tree classifier to predict age-related conditions.
    - **Ensembles – Bagging**: The bagging algorithm in the code is used with an unpruned decision trees (i.e. max_depth = None) as a base classifier, and 500 of such trees are used. Each tree is trained on a different bootstrap sample of the data. In bagging, the final prediction is made by taking a majority vote (for classification).
    - **Ensembles – AdaBoost**: AdaBoost is used with a stump decision tree (ie max_depth = 1) as the weak learners. There are 500 of these decision stumps used. Weights are assigned to the training instances and are adjusted at every iteration to give more priority to the misclassified instances.
- **KNN**: This algorithm in the code uses 5 neighbors to make predictions and the Minkowski

- distance with p=2 ie Euclidean distance. The features are standardized using StandardScaler() from sklearn, and are scaled to have zero mean and unit variance before being fed into the KNN model, to achieve optimal performance with KNN.
    - **Ensembles – Majority Vote**: we combined the predictions of multiple models - a decision tree and KNN. The decision tree used in this ensemble has a maximum depth of 2 and uses entropy as the splitting criterion. The KNN model has the same model specification as in the KNN model. We used a user-defined MajorityVoteClassifier to for majority rule voting.
- **Neural network**: We will run variations with different numbers of hidden layers, numbers of nodes, activation algorithms (tanh, relu), and optimizers (Adam, SGD) to identify the best configuration for this dataset. Weights are randomized when the model is initialized and updated through back propagation.

Model evaluation

- **Accuracy**: We plan to use the ratio of the number of correct predictions to the total number of input samples to measure accuracy, given our dataset will be preprocessed to balance different labels. Accuracy is calculated on both the train and validation dataset to compare results and to identify any overfitting.
- **Precision/Recall/F1-Score**: Precision will tell us the proportion of true positive predictions among all positive predictions. Recall tells us the proportion of true positive predictions among all actual positives. F1-score is the harmonic mean of precision and recall. All of these will be evaluated on validation dataset
- **Confusion Matrix**: A confusion matrix can be used to understand the true positive, true negative, false positive, and false negative, as well as to gain insights into the type of errors made by the model.
- **10-Fold Cross-Validation**: We will divide the valuation dataset into 10 parts, using 9 parts for training and 1 part for testing. This process will be repeated 10 times with each part being used as a test set once. This helps understand how the model will perform on unseen data, as well as fine-tune the model performance given set of hyperparameters.

Berkeley
UNIVERSITY OF CALIFORNIA

To evaluate our results we plan to use the following performance metrics:

- For accuracy, we will measure this by comparing the number of correct predictions to the total input samples. We will preprocess the dataset to balance labels and calculate accuracy on both the train and validation datasets to identify overfitting.
- We are also evaluating precision, recall, and F-1 score. Precision will help us measure true positive predictions among positive predictions, recall will help us measure true positive predictions among actual positives, and the F1-score is the harmonic mean of precision and recall. These metrics will be evaluated on the validation dataset.
- In addition to the evaluation metrics, we will also use a confusion matrix. The confusion matrix allows us to gain deeper insights into the model's performance by understanding true positive, true negative, false positive, and false negative predictions. By analyzing the confusion matrix, we can identify the specific types of errors made by our model and take necessary actions to improve its performance.
- Lastly, the 10-fold cross-validation technique. This technique will help us assess how our model will perform on unseen data and allows us to fine-tune its performance with a given set of hyperparameters. We plan to divide the validation dataset into 10 parts. During each iteration, we use 9 parts for training our model and reserve 1 part for testing. This process is repeated 10

- times, with each part being used as a test set once. By doing so, we can evaluate our model's performance across different subsets of the data and gain a more robust understanding of its capabilities.

  This concludes our baseline presentation and we are excited to dig more into this topic. Thank you.