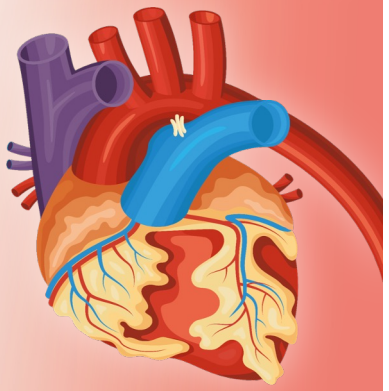
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO



Heart Disease Prediction

Project II – Artificial Intelligence

Group 36:
Catarina Barbosa – up202004898@fc.up.pt
Francisca Andrade – up202005977@fc.up.pt
Marcos Ferreira – up201800177@fe.up.pt

Faculdade de Engenharia da Universidade do Porto

Problem Specification

Heart diseases are the leading cause of death worldwide, account for millions of deaths per year. The detection of a heart problem can be crucial in prevention and early treatment of the disease before a heart attack, stroke and/or other serious problems occurs.

The objective of this project is to develop an artificial intelligence model that can predict whether a person has a heart disease or not.

To develop a REALISTIC we analyze, explore, clean and process the data available as well as train different models with different configuration to compare their performances and choose the most fitting one.

Problem Specification

The dataset used contains the following information:

- The BMI of the person
- If the person smokes and/or drinks alcohol
- If the person had a stroke previously
- A physical and mental health score
- Basic info (age, sex, race)
- Known conditions such as (asthma, kidney problems, diabetes, cancer)
- Sleeping time
- If the person practices physical activities
- A general health score
- If a person has a weaker/slower walk

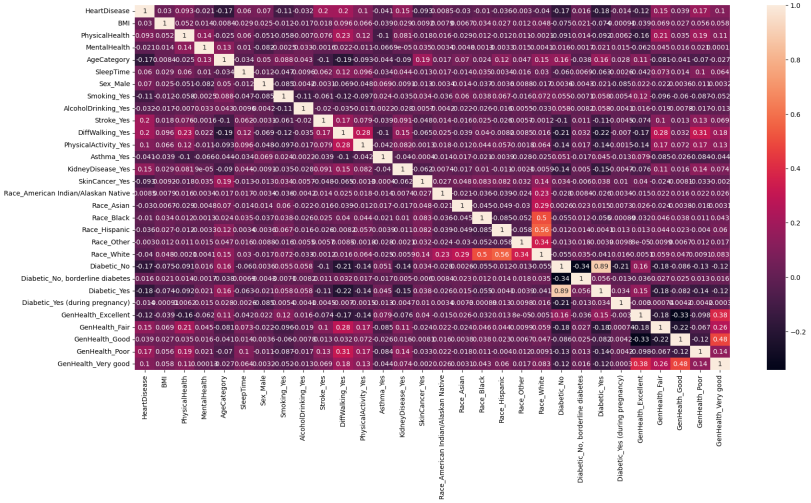
Data Processing

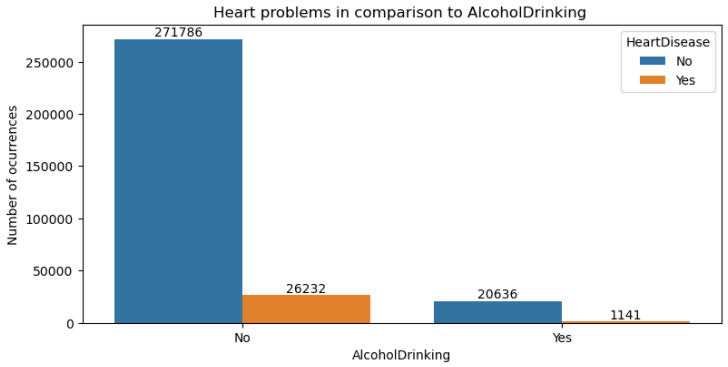
- Data analysis:
  - For each feature and the label, we analyzed the following information:
    - The distinct values for the feature and their occurrences (number of occurrences and percentages);
    - If there were any missing values (none found);
  - Comparison between label and features
  - Analysis of the correlation matrix
  - To adjust some features for the models we had to encode some of them:
    - Label encoding for the label (change from "yes/no" to "1/0");
    - One hot encoding for variables with "yes/no" or "yes/maybe/no" (as an example) for features;
    - Ordinal encoding for "age category) - while it might not be the best idea we wanted to experiment and learn how to do it.
  - This generated features that we once again correlated, therefore we removed them (for example "Stroke no" and "Stroke yes", we dropped "Stroke no").

```
INFOR FOR: AlcoholDrinking

Basic info
Type of data: object
Disntinct values: ['No' 'Yes']
Number of disntinct values:2
Number of occurences for values:
No     298018
Yes     21777
Name: AlcoholDrinking, dtype: int64
Percentage of occurence for values:
No     93.190325
Yes     6.809675
Name: AlcoholDrinking, dtype: float64
number of missing values: 0
```

Data for alcohol, drinking





Graph for alcohol drinking

```
Statistics


Minimum Value: 0.0
Maximum value: 30.0
Range of values: 30.0
Number of zeros: 205401
Percentage of zeros: 64.22895917697275%
Mean value: 3.898366140808956
Meadian value:0.0
Standard deviation: 7.955235218943607
Coefficient of variation: 2.0406588123333136
Meadian Absolute Deviation (MAD): 5.430809568785669
Skewness: 2.331111549136165
Sum of column: 1246678.0
Variance: 63.28576738872075
Is variable monotonic: False
Percentiles:
     5th pertentile: 0.0
     25th pertentile: 0.0
     50th pertentile: 0.0
     75th pertentile: 3.0
     95th pertentile: 30.0
```

Statistic data for mental health score

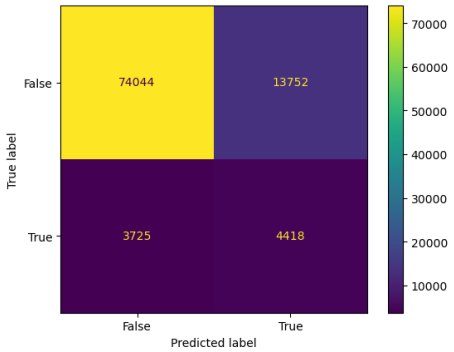
Models

- Divided the data into a "70/30" division for train data and test data respectively.
- Accuracy was not a good measure since we had a lot more examples of "no heart disease" than other wise, so we analyzed the ROC AUC
- We tested 6 different models (we tried using different models to learn different approaches):
  - Naïve Bayes: a simple a quick model to serve as comparison base;
  - Logistic regression;
  - KNN;
  - Decision tree;
  - Random forest;
  - Neural networks;
- For the first run we performed a simple run of the models. As we can verify the results weren't that good. It was expected since the dataset is imbalanced.

Naïve Bayes

	precision	recall	f1-score	support
0	0.95	0.84	0.89	87796
1	0.24	0.54	0.34	8143
accuracy			0.82	95939
macro avg	0.60	0.69	0.62	95939
weighted avg	0.89	0.82	0.85	95939

ROC AUR: 0.6929580237155332



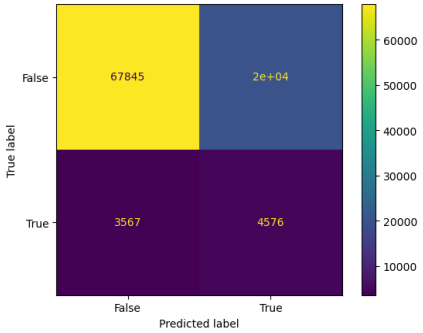
OverSampling

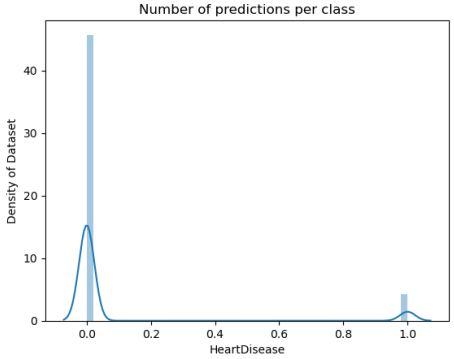
- Oversampling with SMOTE
  - While SMOTE helps fix the problem of imbalance in the dataset, it creates new entries which can be "strange" or make the noise in the data worse.

KNN

	precision	recall	f1-score	support
0	0.95	0.77	0.85	87796
1	0.19	0.56	0.28	8143
accuracy			0.75	95939
macro avg	0.57	0.67	0.57	95939
weighted avg	0.89	0.75	0.80	95939

ROC AUR: 0.6673561772180535



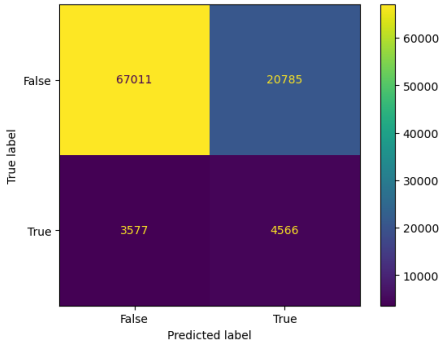


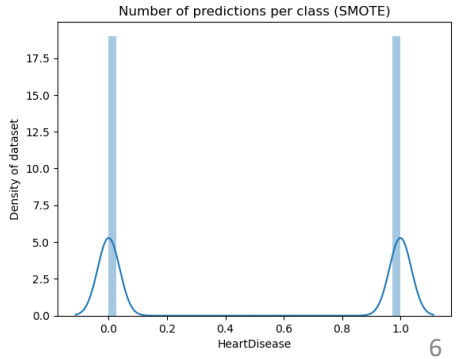
- Oversampling with ADASYN
  - ADASYN tries to improve on SMOTE by creating examples of the minority class while also trying to solve SMOTE problems by generating examples in areas of the data that are harder to classify.

KNN

	precision	recall	f1-score	support
0	0.95	0.76	0.85	87796
1	0.18	0.56	0.27	8143
accuracy			0.75	95939
macro avg	0.56	0.66	0.56	95939
weighted avg	0.88	0.75	0.80	95939

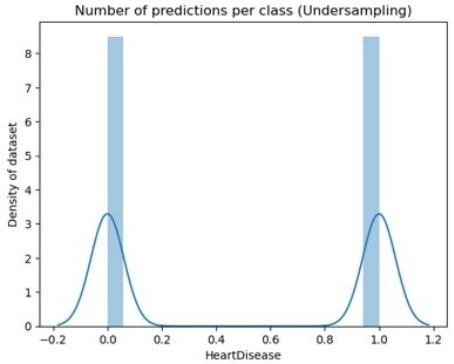
ROC AUR: 0.6619925059939477





Undersampling

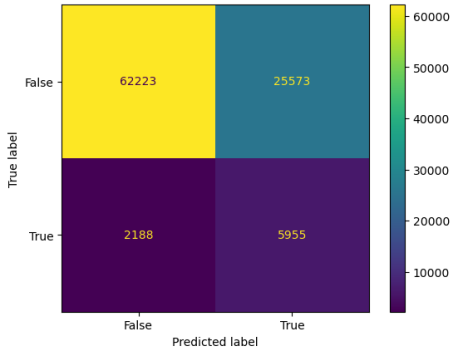
- Since the dataset was large enough, we also tried doing undersampling.
- Undersampling yielded better results, therefore we will continue using it.



KNN

	precision	recall	f1-score	support
0	0.97	0.71	0.82	87796
1	0.19	0.73	0.30	8143
accuracy			0.71	95939
macro avg	0.58	0.72	0.56	95939
weighted avg	0.90	0.71	0.77	95939

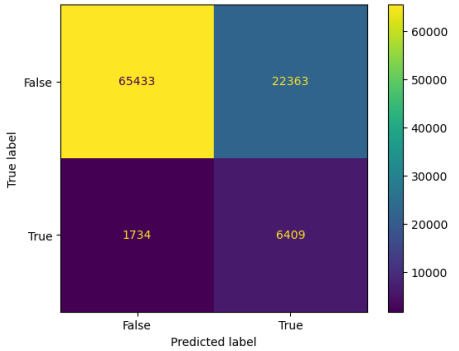
ROC AUR: 0.7200127263246376



Neural networks

	precision	recall	f1-score	support
0	0.97	0.75	0.84	87796
1	0.22	0.79	0.35	8143
accuracy			0.75	95939
macro avg	0.60	0.77	0.60	95939
weighted avg	0.91	0.75	0.80	95939

ROC AUR: 0.7661704453225265



Feature Selection and Extraction

- Feature selection (using a logistic regression to select the features with tests for different number of features selected).
- Feature extraction (using pca).

```
# create the RFE object and set the parameters
cross_logisticModel = LogisticRegression(solver='lbfgs', max_iter=2000)
rfe = RFE(cross_logisticModel, n_features_to_select=10)

#copying the data to not lose it for feature extraction
x_train_fs, y_train_fs = x_train, y_train

# fit the RFE object to the data
rfe.fit(x_train_fs, y_train_fs)

# print the selected features
print("Selected Features: ", x_train_fs.columns[rfe.support_])
```

✓ 56.1s Python

```
Selected Features:  Index(['Sex_Male', 'Stroke_Yes', 'DiffWalking_Yes', 'KidneyDisease_Yes',
       'Race_White', 'Diabetic_No', 'Diabetic_Yes (during pregnancy)',
       'GenHealth_Excellent', 'GenHealth_Poor', 'GenHealth_Very good'],
      dtype='object')
```

Cross-validation

- Feature selection or extraction had no significant impact. Because of this, we will continue to use the original features.
- To try and further enhance the results we will be trying cross-validation (while using undersampling).

Results and conclusion

	Initial		Final	
	Recall	**Roc Aur**	**Recall**	**Roc Aur**
Naïve Bayes	0.690	0.692	0.641	0.796
Logistic regression	0.550	0.551	0.776	0.839
KNN	0.530	0.534	0.714	0.770
Decision Tree	0.590	0.590	0.663	0.671
Random forest	0.550	0.559	0.765	0.810
Neural Networks	0.530	0.529	0.797	0.840

In this project, we mastered training and testing machine learning models. We analyzed data, evaluated models, handled data imbalance, selected features, and used cross-validation. This project enhanced our skills in training and testing models accurately and efficiently.

Tools and References

- Tools

The project will be done in python using a Jupyter Notebook environment. In addition, the following libraries will be used:

  - Numpy – Array and Matrices processing
  - Pandas – Data analysis and manipulation
  - Matplotlib – Generation of graphics and tables
  - Seaborn - Statistical data visualization
  - Sklearn – Implementation of the models and metrics for them

- References
  - https://towardsdatascience.com/predictive-modeling-picking-the-best-model-69ad407e1ee7
  - https://www.sas.com/en_gb/insights/articles/analytics/a-guide-to-predictive-analytics-and-machine-learning.html
  - https://www.kaggle.com/learn/intro-to-machine-learning