# Heart Disease Prediction App - Capstone Project

**Introduction**

Cardiovascular diseases (CVDs) is a disorder group that affects the heart and blood vessels that some of them may result in heart attacks and strokes that are acute events and very letal.
A research from the World Health Organization (WHO) points out that every year, almost 18 million people lose their lives to CVDs worldwide, being the main cause of death globally.

Knowing the severity of the disease, and the difficulties of getting assumptions of the risk of having the disease, there are some clinical features that may help identifying the disease, such as cholesterol and blood pressure. But it's not only about these ones that can truly tell with a good certainty.

Another problem is about the massive amount of people under superficial diagnoses and mostly discarded because of superficial decisions. It's very risky given the lethality and how sudden and acute these diseases act.

So, wouldn't it be helpful having a model that can identify possible risks of having the disease given some clinical results of many patients? Maybe just a program that returns a prognostic with a possibility of the patient having the illness?

**Development**

Building a model with such capacity is a little complex due to its difficulty and risk as we are handling lives. But we can start with the basis and improve the model as we get the first quick-wins and good results.

For instance, there is a dataset containing a few observations of clinical exams on patients focusing on heart disease diagnoses which was initially separated into 5 datasets from different sources, and now it’s the largest one for research purposes, containing 11 common features and 918 observations and there are the details:

## Source

- Cleveland: 303 observations
- Hungarian: 294 observations
- Switzerland: 123 observations
- Long Beach VA: 200 observations
- Stalog (Heart) Data Set: 270 observations

## Attribute Information

1. Age: age of the patient [years]
2. Sex: sex of the patient [M: Male, F: Female]
3. ChestPainType: chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
4. RestingBP: resting blood pressure [mm Hg]
5. Cholesterol: serum cholesterol [mm/dl]
6. FastingBS: fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
7. RestingECG: resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
8. MaxHR: maximum heart rate achieved [Numeric value between 60 and 202]
9. ExerciseAngina: exercise-induced angina [Y: Yes, N: No]
10. Oldpeak: oldpeak = ST [Numeric value measured in depression]
11. ST_Slope: the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
12. HeartDisease: output class [1: heart disease, 0: Normal]

The data collected for this study was gathered on Kaggle, a data science platform with many datasets for data challenges and sharing knowledge.
Every dataset used can be found under the Index of heart disease datasets from UCI Machine Learning Repository on the following link:
https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/

For analysis, training and choosing the best model, the data was processed in a library called Pycaret, which helps a lot with many tasks. These processes and the results will be shown and explained during the development.

**Overview**

Dataset statistics

Dataset statistics

| | |
|---|---|
| **Number of variables** | 12 |
| **Number of observations** | 918 |
| **Missing cells** | 0 |
| **Missing cells (%)** | 0.0% |
| **Duplicate rows** | 0 |
| **Duplicate rows (%)** | 0.0% |
| **Total size in memory** | 86.2 KiB |
| **Average record size in memory** | 96.1 B |

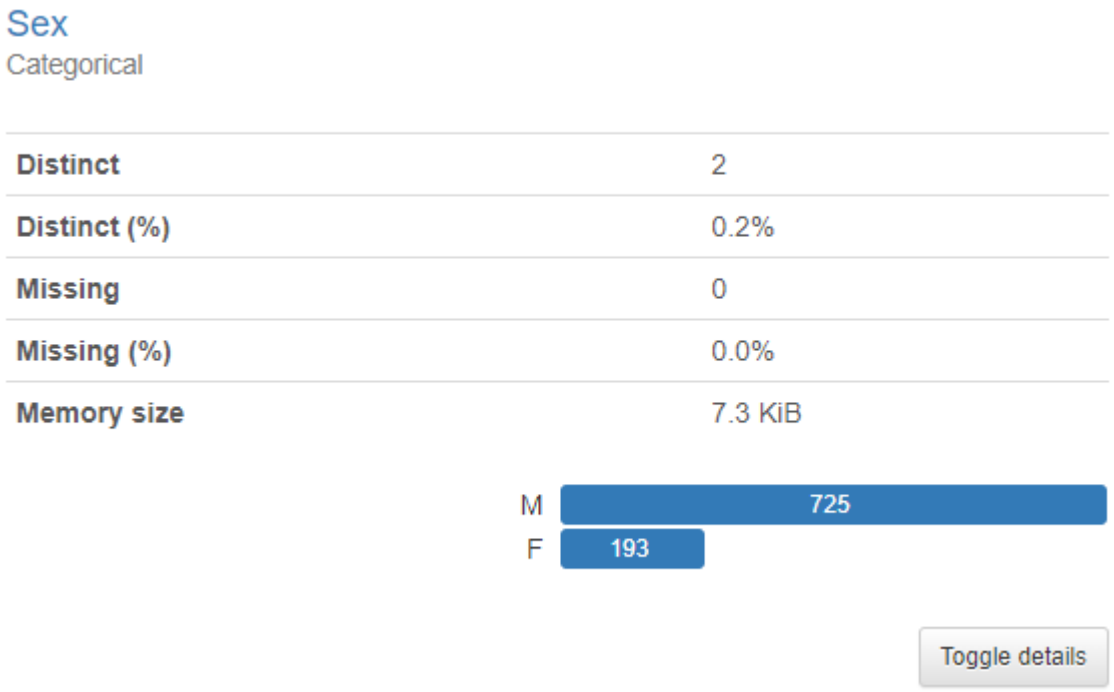
Variable types

| | |
|---|---|
| **Numeric** | 5 |
| **Categorical** | 6 |
| **Boolean** | 1 |

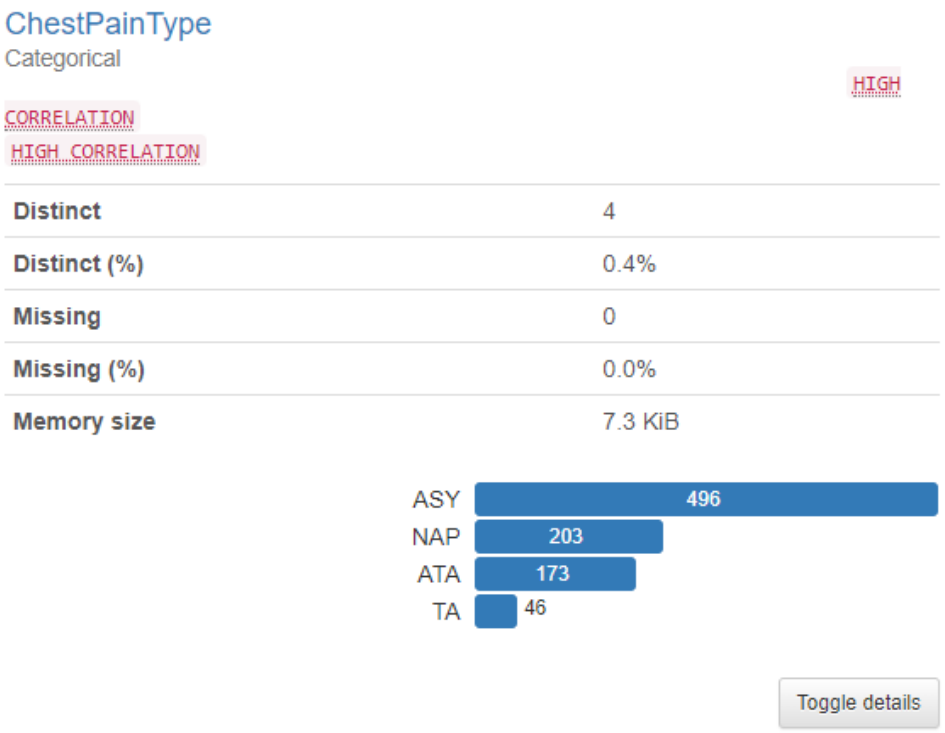
The dataset itself has 918 observations and 12 variables as mentioned in the beginning. It also has no duplicate data, saving from some preprocessing and cleaning steps. Variable Types well divided, 5 Numerics, 6 Categorical and 1 Boolean/Binary.

Opening the variable visualizations, there are some relevant variables and may need cleaning and preprocessing such as:

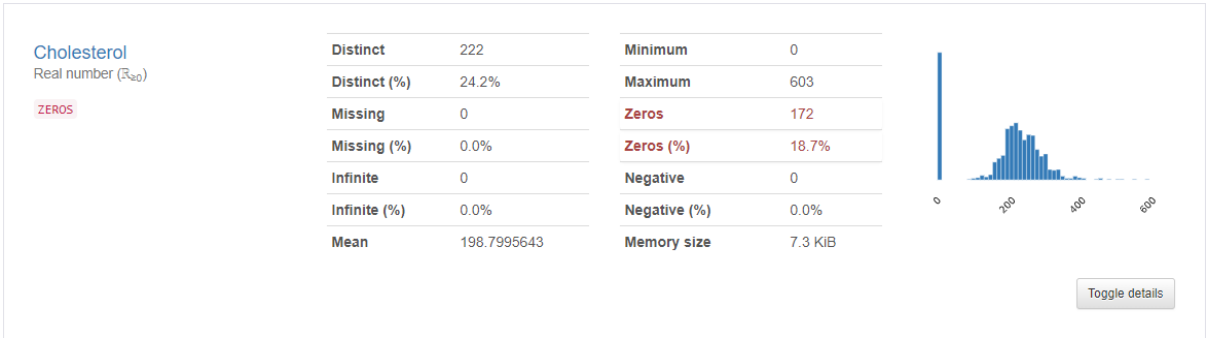
The Sex variable shows that our dataset has Male observations as the majority.




Chest Pain Type with ASY (Asymptomatic) in the majority, balanced NAP and ATA, and a few TA.




About cholesterol, rather clean some elements that are equal to zero, as there is no sense of some person having a cholesterol level in zero, and so in this high number of observations. In this case a good strategy would be replacing de zeros for the median of the dataset. There are some values very high too, but as they are only a few compared to the overall, they should be considered as outliers and remove them.

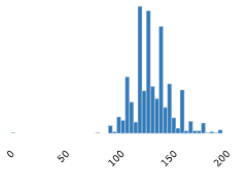



The RestingBP variable also has a similar case to Cholesterol. Having the blood pressure at zero is impossible, but in this case are only some observations and can be put aside as outliers and removed from the dataset.

RestingBP

Real number ($\mathbb{R}_{\geq 0}$)

| | | | |
|---|---|---|---|
| **Distinct** | 67 | **Minimum** | 0 |
| **Distinct (%)** | 7.3% | **Maximum** | 200 |
| **Missing** | 0 | **Zeros** | 1 |
| **Missing (%)** | 0.0% | **Zeros (%)** | 0.1% |
| **Infinite** | 0 | **Negative** | 0 |
| **Infinite (%)** | 0.0% | **Negative (%)** | 0.0% |
| **Mean** | 132.3965142 | **Memory size** | 7.3 KiB |




Toggle details

And finally, our target variable: Does the patient have a heart disease?
The dataset target is well balanced, but not perfect.

## HeartDisease

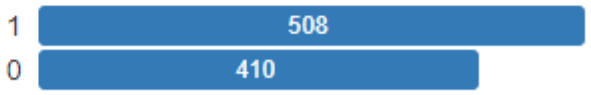
Categorical

HIGH

CORRELATION

HIGH CORRELATION

| | |
|---|---|
| **Distinct** | 2 |
| **Distinct (%)** | 0.2% |
| **Missing** | 0 |
| **Missing (%)** | 0.0% |
| **Memory size** | 7.3 KiB |




Toggle details

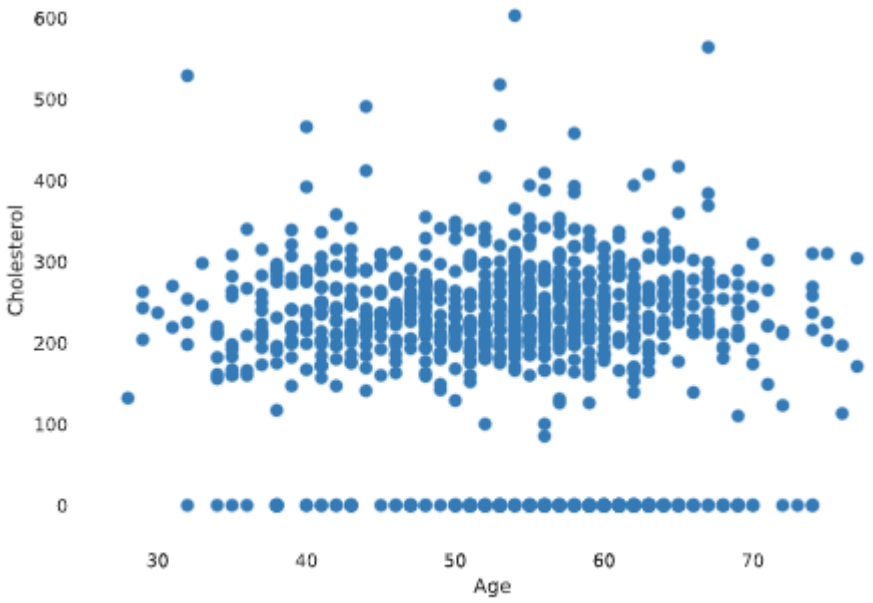
It’s worth to reinforce that only some of the most relevant variables were shown, the other ones were cleaned and preprocessed.

**Interactions**

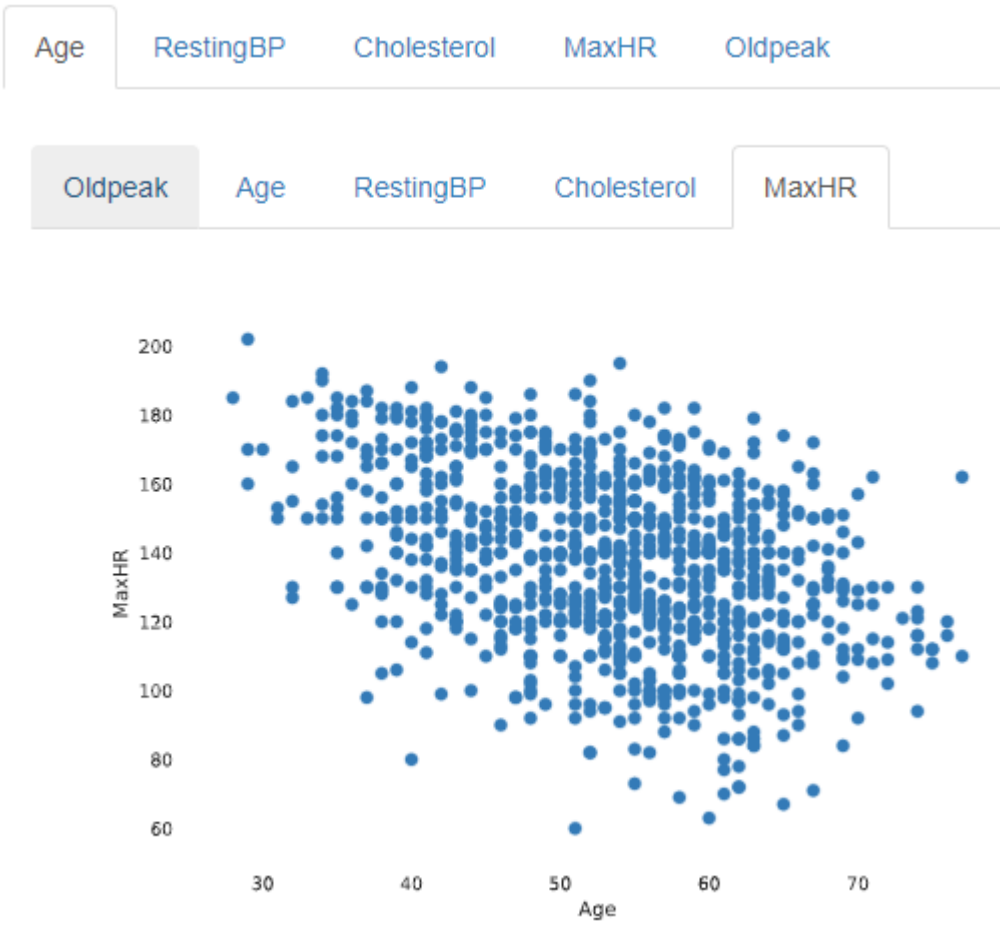
Age x Cholesterol

Age | RestingBP | Cholesterol | MaxHR | Oldpeak

Oldpeak | Age | RestingBP | Cholesterol | MaxHR

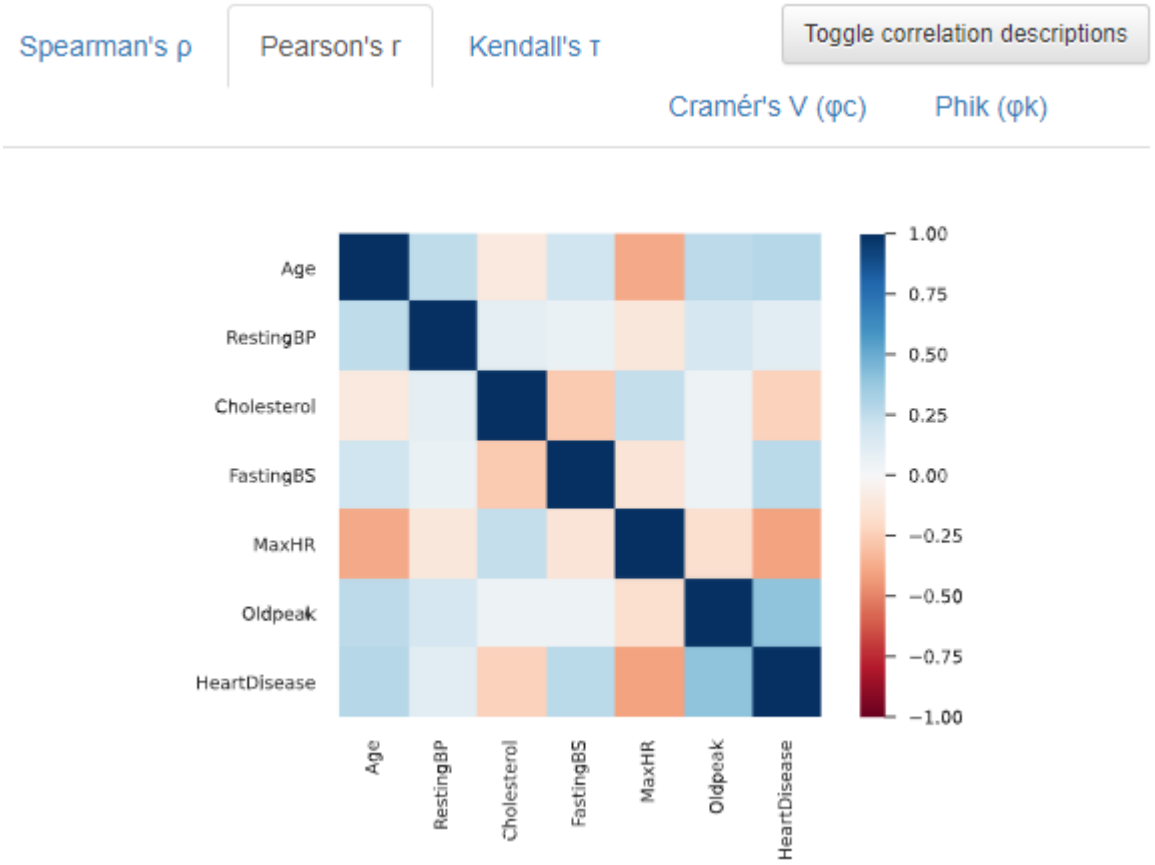


Age x MaxHR




Most of the interaction between two variables results in almost the same pattern. No evidence of high correlation as the Pearson's correlation heatmap shows below.

Correlations




Not a single variable with more than 0.5 or less than 0.5 in correlation.

In resume only two variables passed by a cleaning and outlier filtering:

Cholesterol - Zeros replaced by the median and removed values above 400.
RestingBP - Removed values below 80.

**Choosing and Building a model**

The Pycaret library is capable of taking a given dataset and iterating over some known models.

There are some function that may help saving some time with preprocessing and abstracting several steps into high-level functions like:

**Setup - Initializing function**

A function that abstracts basic tasks for machine learning experiments. A large setup of pre-processing features that helps in leveraging solutions to advanced levels.
The first analysis of the features was a great example of how complete it is since the beginning.

Another advantage is how large is the covering of preprocessing steps. With a quick analysis of the dataset, the function itself builds a report pointing some key processing rules to consider while building and testing a model. Here is an example:

```
from pycaret.classification import *
#intialize the setup
exp_clf = setup(heart, target = 'HeartDisease')
```

| | Description | Value |
|---|---|---|
| 0 | session_id | 8220 |
| 1 | Target | HeartDisease |
| 2 | Target Type | Binary |
| 3 | Label Encoded | None |
| 4 | Original Data | (918, 12) |
| 5 | Missing Values | False |
| 6 | Numeric Features | 5 |
| 7 | Categorical Features | 6 |
| 8 | Ordinal Features | False |
| 9 | High Cardinality Features | False |
| 10 | High Cardinality Method | None |
| 11 | Transformed Train Set | (642, 18) |
| 12 | Transformed Test Set | (276, 18) |
| 13 | Shuffle Train-Test | True |
| 14 | Stratify Train-Test | False |
| 15 | Fold Generator | StratifiedKFold |
| 16 | Fold Number | 10 |
| 17 | CPU Jobs | -1 |

| | | |
|---|---|---|
| **18** | Use GPU | False |
| **19** | Log Experiment | False |
| **20** | Experiment Name | clf-default-name |
| **21** | USI | d3ee |
| **22** | Imputation Type | simple |
| **23** | Iterative Imputation Iteration | None |
| **24** | Numeric Imputer | mean |
| **25** | Iterative Imputation Numeric Model | None |
| **26** | Categorical Imputer | constant |
| **27** | Iterative Imputation Categorical Model | None |
| **28** | Unknown Categoricals Handling | least_frequent |
| **29** | Normalize | False |
| **30** | Normalize Method | None |
| **31** | Transformation | False |
| **32** | Transformation Method | None |
| **33** | PCA | False |
| **34** | PCA Method | None |
| **35** | PCA Components | None |
| **36** | Ignore Low Variance | False |
| **37** | Combine Rare Levels | False |
| **38** | Rare Level Threshold | None |
| **39** | Numeric Binning | False |
| **40** | Remove Outliers | False |
| **41** | Outliers Threshold | None |
| **42** | Remove Multicollinearity | False |
| **43** | Multicollinearity Threshold | None |

| | | |
|---|---|---|
| 44 | Remove Perfect Collinearity | True |
| 45 | Clustering | False |
| 46 | Clustering Iteration | None |
| 47 | Polynomial Features | False |
| 48 | Polynomial Degree | None |
| 49 | Trigonometry Features | False |
| 50 | Polynomial Threshold | None |
| 51 | Group Features | False |
| 52 | Feature Selection | False |
| 53 | Feature Selection Method | classic |
| 54 | Features Selection Threshold | None |
| 55 | Feature Interaction | False |
| 56 | Feature Ratio | False |
| 57 | Interaction Threshold | None |
| 58 | Fix Imbalance | False |
| 59 | Fix Imbalance Method | SMOTE |

**Sample**

A function to make the well-known train test split task, just pass it a dataset and a decimal ratio and it's done.

```
data = dataset.sample(frac=0.8, random_state=42).reset_index(drop=True)
data_unseen = dataset.drop(data.index).reset_index(drop=True)
```

In the Heart dataset, a 0.8 ratio was applied.

**Compare Models**

Once the train and test data are preprocessed, cleaned and splitted, it's time to explore models for solving the main problem. The compare_models function gathers the dataset and

iterates over a batch of stored models and returns a benchmark table ranking the models by its main evaluation metrics, by default accuracy.

However, accuracy wouldn't be any useful or make any sense. In this case better rank the models by Recall. But… Why recall?
Going back for the introduction, as the solution proposes to deal with lives and generating prognostics, it's essential to consider avoiding false negatives.
In medicine prognosing, specially on Cardiovascular diseases, a false negative implies on declaring to a patient that he doesn't have any CVD, but he does have it. The problem is that the diagnoses weren't precise enough to catch it or insufficient for a complete checkup.
So it's highly preferable to have false-positives rather than false negatives, this is why the Recall score must be used for boosting the model.

```
compare_models(sort='Recall')
```

| | Model | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC | TT (Sec) |
|---|---|---|---|---|---|---|---|---|---|
| rf | Random Forest Classifier | 0.8599 | 0.9172 | 0.9008 | 0.8558 | 0.8763 | 0.7148 | 0.7198 | 0.512 |
| catboost | CatBoost Classifier | 0.8692 | 0.9263 | 0.8980 | 0.8722 | 0.8824 | 0.7348 | 0.7407 | 1.654 |
| gbc | Gradient Boosting Classifier | 0.8630 | 0.9173 | 0.8895 | 0.8703 | 0.8770 | 0.7221 | 0.7287 | 0.122 |
| ridge | Ridge Classifier | 0.8582 | 0.0000 | 0.8837 | 0.8665 | 0.8726 | 0.7126 | 0.7180 | 0.013 |
| lda | Linear Discriminant Analysis | 0.8566 | 0.9227 | 0.8837 | 0.8637 | 0.8713 | 0.7094 | 0.7146 | 0.016 |
| lightgbm | Light Gradient Boosting Machine | 0.8536 | 0.9233 | 0.8810 | 0.8626 | 0.8689 | 0.7030 | 0.7094 | 0.079 |
| xgboost | Extreme Gradient Boosting | 0.8443 | 0.9136 | 0.8781 | 0.8481 | 0.8607 | 0.6839 | 0.6891 | 5.452 |
| lr | Logistic Regression | 0.8519 | 0.9244 | 0.8752 | 0.8638 | 0.8668 | 0.7001 | 0.7059 | 0.421 |
| et | Extra Trees Classifier | 0.8427 | 0.9118 | 0.8698 | 0.8534 | 0.8585 | 0.6813 | 0.6881 | 0.470 |
| nb | Naive Bayes | 0.8473 | 0.9173 | 0.8666 | 0.8643 | 0.8621 | 0.6907 | 0.6971 | 0.014 |
| ada | Ada Boost Classifier | 0.8364 | 0.8969 | 0.8524 | 0.8559 | 0.8502 | 0.6695 | 0.6771 | 0.107 |
| dt | Decision Tree Classifier | 0.7990 | 0.7969 | 0.8189 | 0.8201 | 0.8167 | 0.5940 | 0.5988 | 0.016 |
| knn | K Neighbors Classifier | 0.6870 | 0.7327 | 0.7225 | 0.7166 | 0.7156 | 0.3671 | 0.3728 | 0.118 |
| qda | Quadratic Discriminant Analysis | 0.6344 | 0.6302 | 0.6756 | 0.7054 | 0.6219 | 0.2632 | 0.3049 | 0.015 |
| svm | SVM - Linear Kernel | 0.5872 | 0.0000 | 0.4263 | 0.8452 | 0.4716 | 0.2023 | 0.2859 | 0.015 |

Above is the result of the best models ordered by Recall, and the creation of the model instance as the following.

```
model = create_model('catboost', random_state=2202)
```

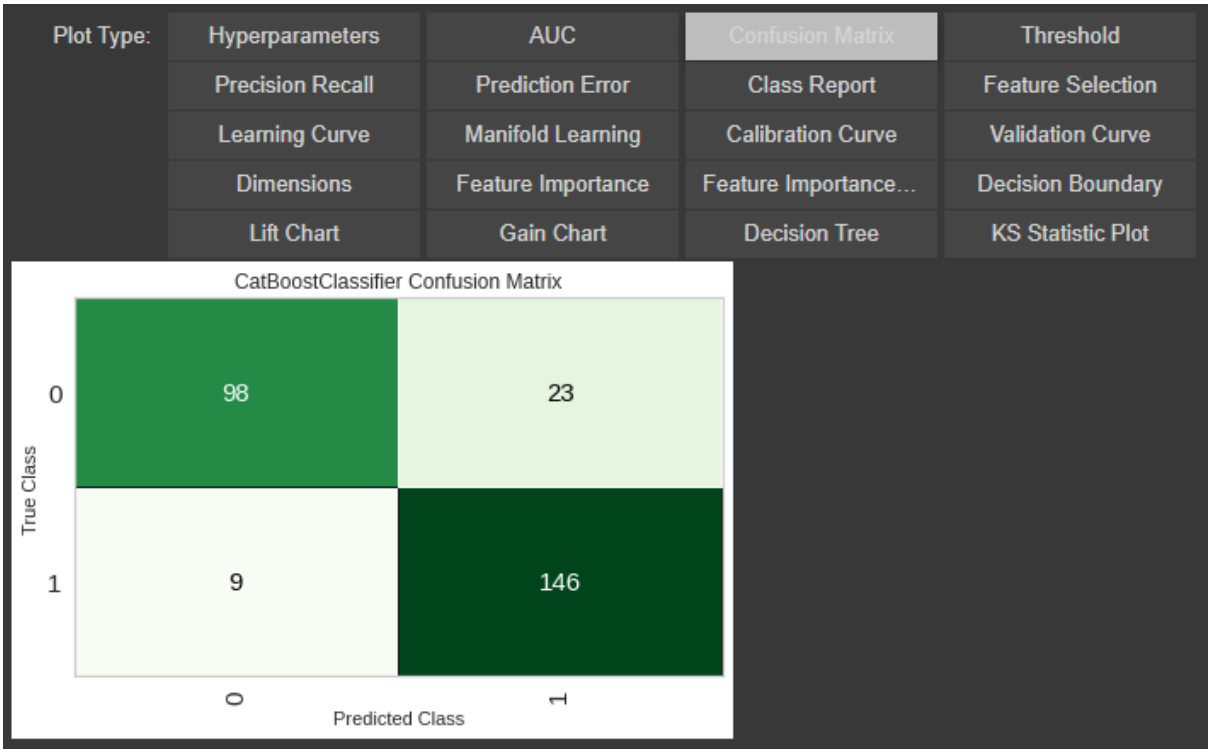
After a set of iterations, the 'catboost' classifier showed the best results under the random state 2202

| | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC |
|---|---|---|---|---|---|---|---|
| 0 | 0.8308 | 0.9109 | 0.9444 | 0.7907 | 0.8608 | 0.6493 | 0.6661 |
| 1 | 0.8923 | 0.9224 | 0.9167 | 0.8919 | 0.9041 | 0.7814 | 0.7817 |
| 2 | 0.9062 | 0.9813 | 0.8857 | 0.9394 | 0.9118 | 0.8119 | 0.8135 |
| 3 | 0.8594 | 0.9468 | 0.8857 | 0.8611 | 0.8732 | 0.7154 | 0.7158 |
| 4 | 0.7969 | 0.9044 | 0.7714 | 0.8438 | 0.8060 | 0.5938 | 0.5964 |
| 5 | 0.8594 | 0.8936 | 0.9714 | 0.8095 | 0.8831 | 0.7103 | 0.7290 |
| 6 | 0.8750 | 0.9005 | 0.9429 | 0.8462 | 0.8919 | 0.7448 | 0.7509 |
| 7 | 0.8594 | 0.9094 | 0.8857 | 0.8611 | 0.8732 | 0.7154 | 0.7158 |
| 8 | 0.9375 | 0.9626 | 0.8857 | 1.0000 | 0.9394 | 0.8754 | 0.8822 |
| 9 | 0.8438 | 0.9395 | 0.8056 | 0.9062 | 0.8529 | 0.6875 | 0.6929 |
| Mean | 0.8661 | 0.9271 | 0.8895 | 0.8750 | 0.8796 | 0.7285 | 0.7344 |
| SD | 0.0376 | 0.0277 | 0.0586 | 0.0588 | 0.0346 | 0.0764 | 0.0754 |

The model is now built using the catboost classifier with a Recall mean of 0.8895.
Now, how is it evaluated?

The library also offers many functions for evaluation steps, such as AUC, Confusion Matrix, Learning Curve and others...
The objective function is about elevating the Recall, so for visualizing it better, a Confusion Matrix should be enough to have an overview.

```
evaluate_model(model)
```



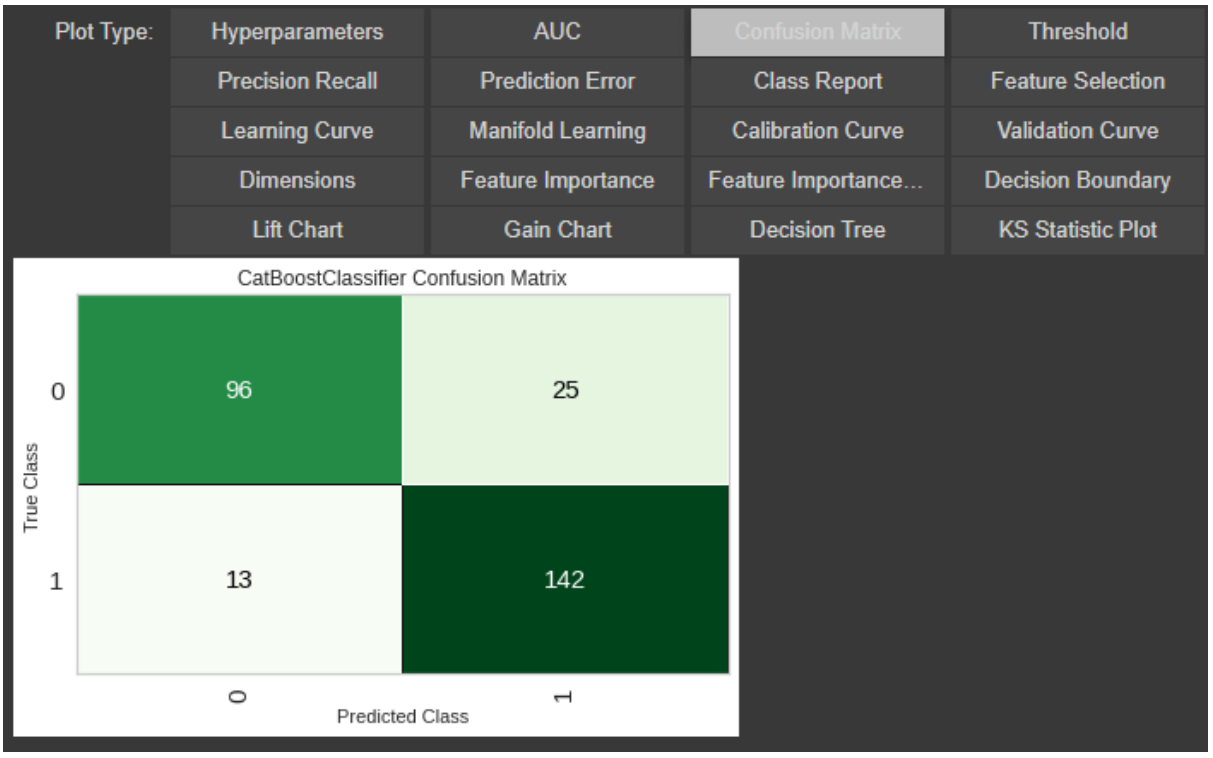

Applying the Recall formula

Recall = TP/ (TP + FN) =  0.9859…
Over 276 observations, only 9 were False Negatives.

Giving a try for tuning the model, there is a tuning function which tries to adjust some hyperparameters in order to get better results that in fact failed to it and leading an assumption that the catboost model would perform better as it is without tuning it. See below the results.

```
tuned_model = tune_model(model, optimize='Recall')
```

| | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC |
|---|---|---|---|---|---|---|---|
| 0 | 0.8462 | 0.9148 | 0.9167 | 0.8250 | 0.8684 | 0.6845 | 0.6900 |
| 1 | 0.8923 | 0.8994 | 0.9444 | 0.8718 | 0.9067 | 0.7799 | 0.7834 |
| 2 | 0.9219 | 0.9655 | 0.9143 | 0.9412 | 0.9275 | 0.8428 | 0.8432 |
| 3 | 0.8906 | 0.9251 | 0.9143 | 0.8889 | 0.9014 | 0.7787 | 0.7790 |
| 4 | 0.8125 | 0.9084 | 0.8000 | 0.8485 | 0.8235 | 0.6239 | 0.6251 |
| 5 | 0.8594 | 0.9113 | 0.9143 | 0.8421 | 0.8767 | 0.7137 | 0.7170 |
| 6 | 0.8125 | 0.8956 | 0.9429 | 0.7674 | 0.8462 | 0.6125 | 0.6340 |
| 7 | 0.8125 | 0.9044 | 0.8571 | 0.8108 | 0.8333 | 0.6194 | 0.6207 |
| 8 | 0.8750 | 0.9537 | 0.8857 | 0.8857 | 0.8857 | 0.7478 | 0.7478 |
| 9 | 0.7969 | 0.8929 | 0.8056 | 0.8286 | 0.8169 | 0.5889 | 0.5892 |
| Mean | 0.8520 | 0.9171 | 0.8895 | 0.8510 | 0.8686 | 0.6992 | 0.7029 |
| SD | 0.0404 | 0.0232 | 0.0496 | 0.0459 | 0.0358 | 0.0824 | 0.0805 |




Tuning the model resulted in a few more False Negatives observations, from 9 to 13, of which the additional 4 were moved from the True Labels. A little worsening in the model. Given those results, better keep the model without tuning.

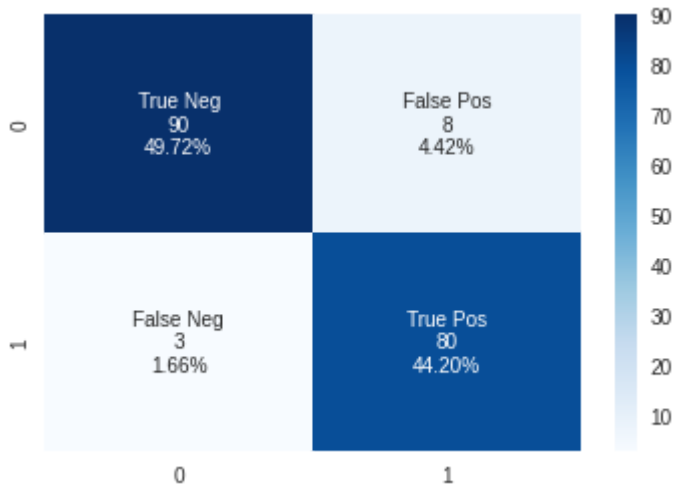
**Predicting and testing**

Now it's time to test it with unseen data, the same data that was separated from the beginning in order to avoid biasing or overfitting.

```
unseen_predictions = predict_model(model, data=data_unseen)
```

| | Age | Sex | ChestPainType | RestingBP | Cholesterol | FastingBS | RestingECG | MaxHR | ExerciseAngina | Oldpeak | ST_Slope | HeartDisease | Label | Score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 54 | M | ASY | 122 | 286 | 0 | LVH | 116 | Y | 3.2 | Flat | 1 | 1 | 0.9626 |
| **1** | 57 | M | ASY | 152 | 274 | 0 | Normal | 88 | Y | 1.2 | Flat | 1 | 1 | 0.9597 |
| **2** | 65 | F | NAP | 160 | 360 | 0 | LVH | 151 | N | 0.8 | Up | 0 | 0 | 0.9169 |
| **3** | 54 | M | NAP | 125 | 273 | 0 | LVH | 152 | N | 0.5 | Down | 0 | 0 | 0.7189 |
| **4** | 54 | F | NAP | 160 | 201 | 0 | Normal | 163 | N | 0.0 | Up | 0 | 0 | 0.9789 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **176** | 45 | M | TA | 110 | 264 | 0 | Normal | 132 | N | 1.2 | Flat | 1 | 1 | 0.8156 |
| **177** | 68 | M | ASY | 144 | 193 | 1 | Normal | 141 | N | 3.4 | Flat | 1 | 1 | 0.9767 |
| **178** | 57 | M | ASY | 130 | 131 | 0 | Normal | 115 | Y | 1.2 | Flat | 1 | 1 | 0.9438 |
| **179** | 57 | F | ATA | 130 | 236 | 0 | LVH | 174 | N | 0.0 | Flat | 1 | 1 | 0.6504 |
| **180** | 38 | M | NAP | 138 | 175 | 0 | Normal | 173 | N | 0.0 | Up | 0 | 0 | 0.9817 |

Here are the results with the unseen data and how the model predicted it followed by a score of certainty of the model.




From 181 observations, 3 of them were False Negatives, resulting in a Recall of 0.9638, a few lower than the test data, but can be interpreted as a good start.

**Conclusions and Considerations**

Reinforcing the idea of building models for prognostics, the main objective is focused on augmenting and encouraging patients to check twice some sets of clinical exams to avoid uncertainties when the approach involves risky illness such as CVDs.

Yet the solution includes analyzing potential risks based on the patterns presented on the training of the model, it must never be interpreted as the real truth, it's only a hypothesis derived from patients who had such illness before.

About the results, even though only 3 from 181 observations were labeled as False Negatives, the real objective is always to aim for the Zero. Anyway, as describing the results it's a good start and opens doors for next steps and advancements in boosting the model.

It's worth remembering that 8 False Positives might be a problem if interpreted as a diagnosis. So the recommendation would be: If the model presented a True Label, that means, a positive diagnosis, that might be a good idea to double check exams to make sure about it.

**How to improve the model:**

There are a few things that may help achieving better results and less uncertainties:

- Gather more observations and include new features or history data
- Get the n top models in Recall and rebuild a benchmark reinforcing hyperparameter tuning
- Balance the distribution of observations in features like Chest Pain Type.
- Review methods for replacing null or invalid data like Cholesterol.

**Extra**

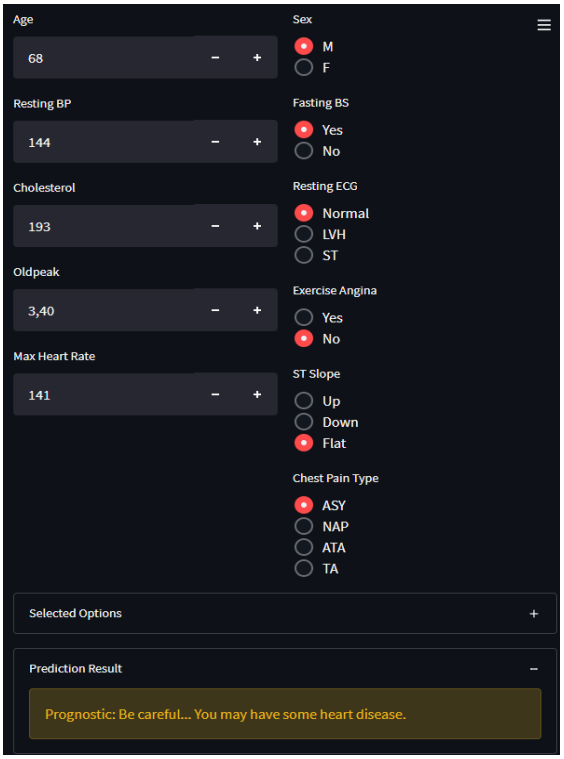
One of the greatest challenges on Data Science and Machine Learning solutions is how to build a product over it and abstract the model results in interfaces making it easier and more usable for non expert users.

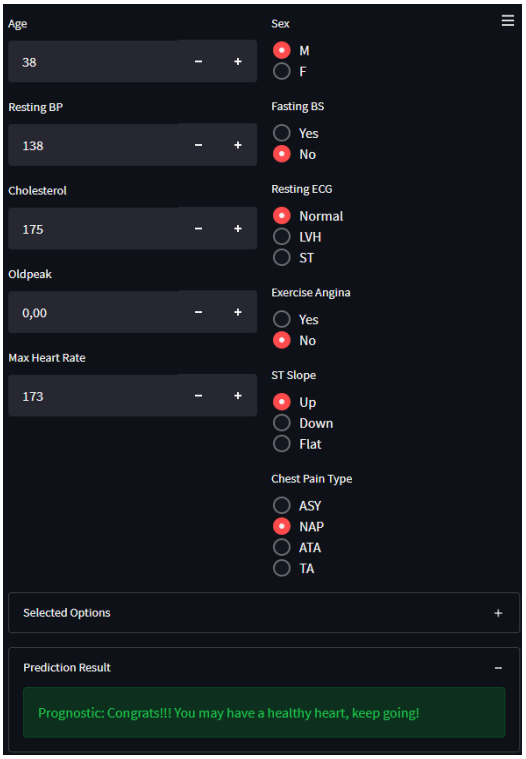
Streamlit is a tool that helps a lot by saving time and easily deploying models. Then building an interface for checking the CVD model wasn't very hard. Just with a simple form frontend offered with few code lines in the library and importing the model, it became possible.

Here is an example of using it with some of the observations presented on the results.

Row 177 and 180 showcase:

The patient number 177 was labeled as a risk of having a disease while 180 seems to not have it so far.




Github repository link:

https://github.com/PyTcheu/Heart-Disease-Prognostic