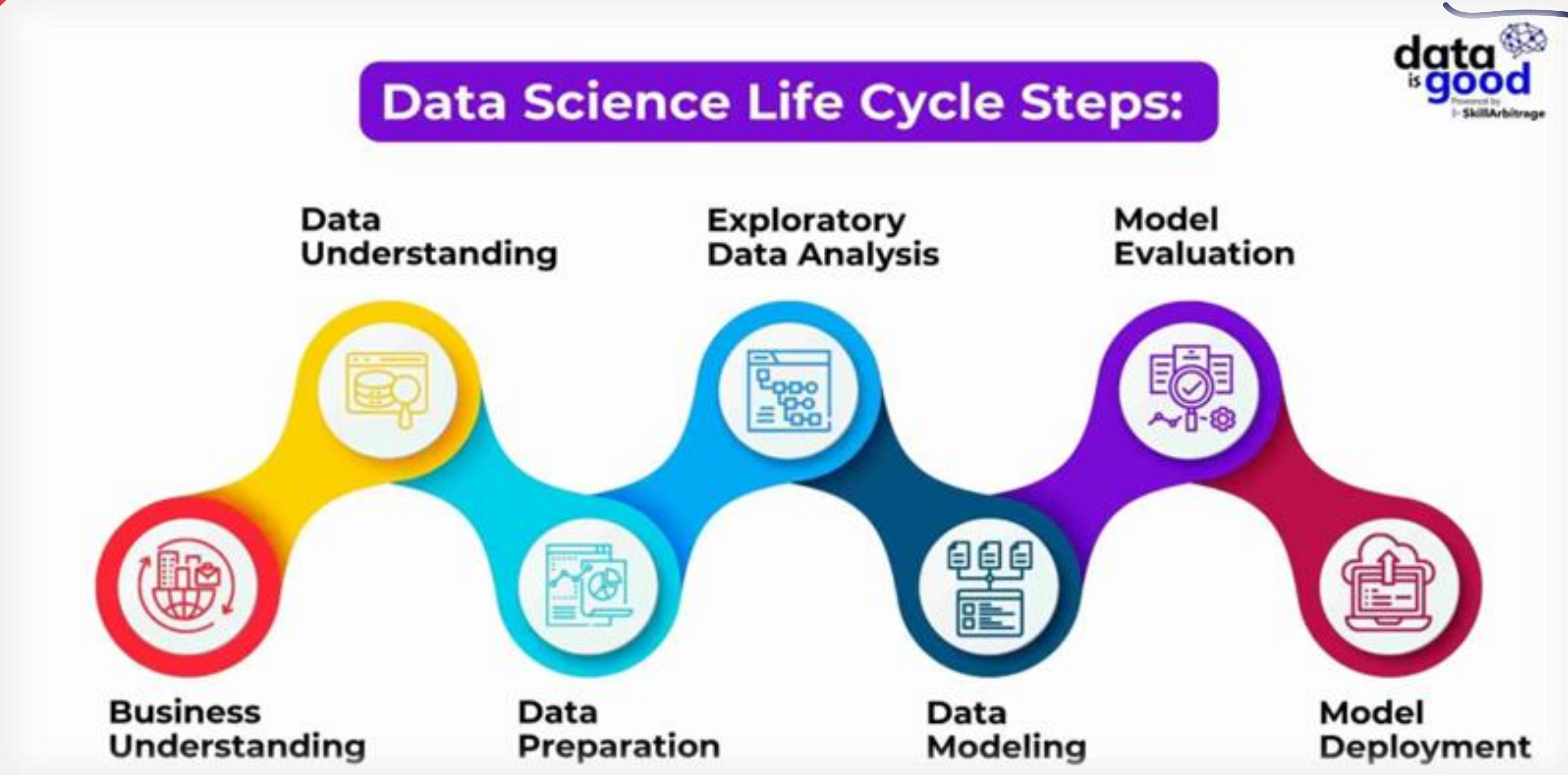
Heart disease:

From Model Building to Deployment

Let’s talk about a special type of heartbreak!

Maximilien Eyengue


Data Science Life Cycle Steps:
Business Understanding
Data Understanding
Data Preparation
Exploratory Data Analysis
Data Modeling
Model Evaluation
Model Deployment

Business Understanding

What?

Disease

According to WHO, heart disease is the leading cause of deaths

33s

So what?

Diagnosis

Early prediction saves lives & reduces costs

Then what?

Objective

Build a tool to accurately identify patients at risk

Data Understanding

```
# download the data
!kaggle datasets download mfarhaannazirkhan/heart-dataset
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /home/maxim-eyengue/.kaggle/kaggle.json'
Dataset URL: https://www.kaggle.com/datasets/mfarhaannazirkhan/heart-dataset
License(s): Attribution 4.0 International (CC BY 4.0)
Downloading heart-dataset.zip to /home/maxim-eyengue/Heart-Disease-App
100%|██████████████████████████████████████| 27.5k/27.5k [00:00<00:00, 216kB/s]
100%|██████████████████████████████████████| 27.5k/27.5k [00:00<00:00, 215kB/s]
```

Five datasets with patient data combined together for heart disease diagnosis

```
# Read the dataframe
df = pd.read_csv("data/raw_merged_heart_dataset.csv")

# Three last rows
df.tail(3)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalachh	exang	oldpeak	slope	ca	thal	target
2178	59	1	3	134	204	0	1	162	0	0.8	2	2	2	0
2179	54	1	1	154	232	0	0	164	0	0.0	2	1	2	0
2180	53	1	0	110	335	0	1	143	1	3.0	1	1	3	0

Data Preparation

```
# Convert resting blood pressure
df.trestbps = pd.to_numeric(df.trestbps, errors = 'coerce')
# Convert cholesterol
df.chol = pd.to_numeric(df.chol, errors = 'coerce')
# Convert maximum heart rate
df.thalachh = pd.to_numeric(df.thalachh, errors = 'coerce')
```

No missing values? Watch out!

```
# For each categorical variable
for cat in categorical:
    # Print the variable and its values
    print(f"{cat} --> {df[cat].unique()}")
```

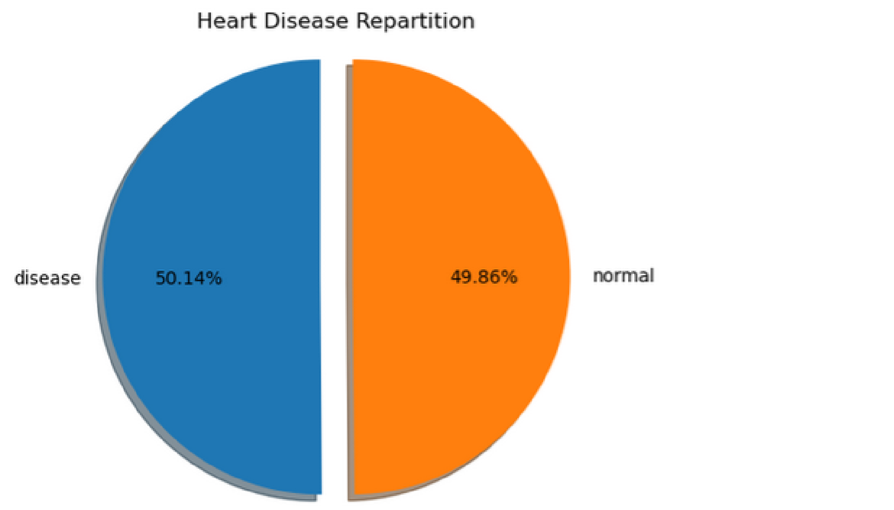
```
sex --> ['male' 'female']
cp --> ['asymptomatic' 'non_anginal_pain' 'atypical_angina' 'typical_angina' nan]
fbs --> ['1' '0' '?']
restecg --> ['0' '1' '2' '?']
exang --> ['0' '1' '?']
slope --> ['0' '2' '1' '?' '3']
ca --> ['0' '2' '1' '3' '4' '?']
thal --> ['1' '2' '3' '0' '?' '6' '7']
```

EDA

There is a low correlation between numerical features

EDA -Target Analysis

```python
# Labels
labs_names = ['disease','normal']
# Target value counts
data_counts = list(df["target"].value_counts())
# Space between pies
pie_space = (0.15, 0)
# Title
plt.title('Heart Disease Repartition')
# Target variable distribution Pie chart
plt.pie(data_counts, explode = pie_space, labels = labs_names , autopct = '%1.2f%%',
        startangle = 90, shadow = True)
# Put title and plot on same axis
plt.axis('equal');
```



The dataset contains approximately the same number of healthy and sick patients

Thus, we can use accuracy score to evaluate our models

EDA – Feature Importance

```
# Apply mutual information columnwise to categorical variables
mi_scores = df_full_train[categorical].apply(mutual_info_y_score)

# Sourt scores in ascending order
mi_scores.sort_values(ascending = False)
```

```
thal       0.14003
cp         0.10439
ca         0.08378
slope      0.08083
exang      0.06419
restecg    0.02542
sex        0.01781
fbs        0.00013
dtype: float64
```

Among categorical features, thalassemia and chest pain information seem to be the most important to determine the disease

The ST segment depression induced by exercise and the maximum heart rate are the most important numerical features

```
# Absolute correlations between numerical columns and target
df_full_train[numerical].corrwith(df_full_train.target).abs().sort_values(ascending = False)
```

```
oldpeak     0.362495
thalachh    0.331442
age         0.168646
chol        0.096872
trestbps    0.095350
dtype: float64
```


Maximilien Eyengue

Data Modeling

Train – Validation – Test split:

```
# Splitting into full train and test
df_full_train, df_test = train_test_split(df, test_size = 0.2, random_state = 42)

# Splitting into train and test
df_train, df_val = train_test_split(df_full_train, test_size = 0.25, random_state = 42)
```

```
# Check datasets sizes after splitting
len(df_train), len(df_val), len(df_test)
```

```
(1083, 361, 361)
```

Target & Features:

```
# Get the target values
y_train = df_train.target.values
y_test = df_test.target.values
y_val = df_val.target.values

# Drop `target` from our data sets
del df_train["target"]
del df_test["target"]
del df_val["target"]
```


Maximilien Eyengue

Data Modeling

Logistic Regression

81%

```python
# Parameter Fine-Tuning
for param in reg_params:
    # Define the model
    model = LogisticRegression(solver = 'liblinear', C = param,
                               max_iter = 1000, random_state = 42)
    # Model fitting
    model.fit(X_train, y_train)
```

Decision Tree

97%

```python
# Decision Tree fine-tuning with maximum-depth
for depth in [10, 15, 20]:
    # Decision Tree model fine-tuning with minimum samples per leaf
    for s in [1, 3, 5, 10, 15, 20, 100, 200, 500]:
        # Initialize the model with a max_depth and min_samples_leaf
        dt = DecisionTreeClassifier(max_depth = depth, min_samples_leaf = s, random_state = 42)
        # Model training
        dt.fit(X_train, y_train)
```

Random Forest

98%

```python
# Fine-tunining with minimum samples per leaf
for s in min_sampl:
    # Fine-tuning with number of estimators
    for n in n_params:
        # Initialize model
        rf = RandomForestClassifier(n_estimators = n,
                                    max_depth = 10,
                                    min_samples_leaf = s,
                                    random_state = 42,
                                    n_jobs = -1)
        # Model training
        rf.fit(X_train, y_train)
```

XG-Boost

97%

```python
# XgBoost fine-tuning with learning rates
for eta_par in eta_values:
    # Model's parameters
    xgb_params = {
        'eta': eta_par,
        'max_depth': 10,
        'min_child_weight': 1,
        'objective': 'binary:logistic',
        'nthread': 8,
        'seed': 1,
        'verbosity': 1,
    }

    # Model training
    model = xgb.train(xgb_params, dtrain,
                      num_boost_round = 200)
```


Maximilien Eyengue

Model Evaluation

```python
# Kfold cross-validation initalization
kfold = KFold(n_splits = n_splits, shuffle = True, random_state = 1)
```

```python
# For each iteration of K-fold split and the pair of indexes generated
for train_idx, val_idx in kfold.split(df_full_train):
    # Select train and validation data
    df_train = df_full_train.iloc[train_idx]
    df_val = df_full_train.iloc[val_idx]

    # Select target variables
    y_train = df_train.target.values
    y_val = df_val.target.values

    # Train model
    One_Hot_encoder, rf = train(df_train, y_train)
    # Make predictions
    y_pred = predict(df_val, One_Hot_encoder, rf)
```

```python
# Print scores' means and standard deviations
print("Validation results:")
print('acc mean = %.2f, acc std = +- %.2f' % (np.mean(scores), np.std(scores)))
```

```
Performing KFold Cross-Validation
Accuracy on fold 0 is 97.23 %.
Accuracy on fold 1 is 97.23 %.
Accuracy on fold 2 is 96.54 %.
Accuracy on fold 3 is 97.92 %.
Accuracy on fold 4 is 97.22 %.
Validation results:
acc mean = 97.23, acc std = +- 0.44
```

With K-Fold cross-validation, the selected model achieved an accuracy of 97.23%, with a standard deviation of 0.44

Model Evaluation

```python
# Optimal parameters values
n_estimators, max_depth, min_samples_leaf = 40, 10, 1
# Optimal random forest model training
One_Hot_encoder, rf = train(df_full_train[categorical + numerical], df_full_train.target,
                            n_estimators = n_estimators, max_depth = max_depth,
                            min_samples_leaf = min_samples_leaf)
# Make predictions
y_pred = predict(df_test, One_Hot_encoder, rf)
# accuracy score
print('Optimal model accuracy = %.2f.' % (100 * (y_pred == y_test).mean()))
```

```
Optimal model accuracy = 95.29.
```

The accuracy of our final model is very good:

95.29%

Maximilien Eyengue

Model Deployment

To easily create an application for our model

To encapsulate the application

For deploying the application to the cloud

Model Deployment

maxim-eyengue@Magic-Maxim: ~/Heart-Disease-App

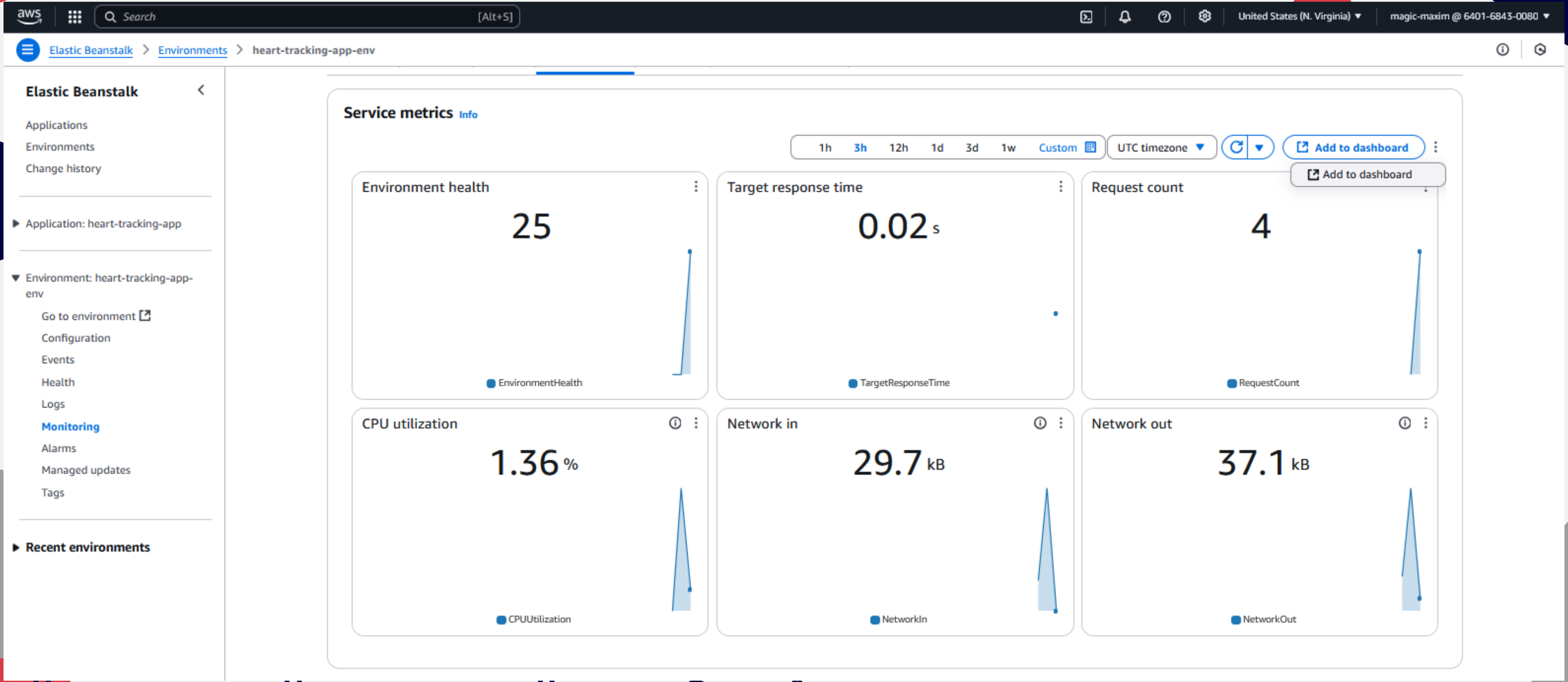
```
(Heart-Disease-App) (base) maxim-eyengue@Magic-Maxim:~/Heart-Disease-App$ eb create heart-tracking-app-env --enable-spot
Creating application version archive "app-ff32-250202_072749941953".
Uploading heart-tracking-app/app-ff32-250202_072749941953.zip to S3. This may take a while.
Upload Complete.
Environment details for: heart-tracking-app-env
  Application name: heart-tracking-app
  Region: us-east-1
  Deployed Version: app-ff32-250202_072749941953
  Environment ID: e-x2dyyppidm
  Platform: arn:aws:elasticbeanstalk:us-east-1::platform/Docker running on 64bit Amazon Linux 2/4.0.7
  Tier: WebServer-Standard-1.0
  CNAME: UNKNOWN
  Updated: 2025-02-02 06:27:57.570000+00:00
Printing Status:
2025-02-02 06:27:55    INFO    createEnvironment is starting.
2025-02-02 06:27:57    INFO    Using elasticbeanstalk-us-east-1-640168430080 as Amazon S3 storage bucket for environment data.
2025-02-02 06:28:19    INFO    Created security group named: sg-01777bc36e21a3314
2025-02-02 06:28:35    INFO    Created target group named: arn:aws:elasticloadbalancing:us-east-1:640168430080:targetgroup/awseb-AWSEB-FQKDGZGKKWEA/82bce1c3d9e21ee0
2025-02-02 06:28:35    INFO    Created security group named: awseb-e-x2dyyppidm-stack-AWSEBSecurityGroup-yx2PoApVRxWh
2025-02-02 06:28:50    INFO    Created Auto Scaling group named: awseb-e-x2dyyppidm-stack-AWSEBAutoScalingGroup-Q3gJgruA8DYe
2025-02-02 06:28:50    INFO    Waiting for EC2 instances to launch. This may take a few minutes.
2025-02-02 06:29:06    INFO    Created Auto Scaling group policy named: arn:aws:autoscaling:us-east-1:640168430080:scalingPolicy:c34e8320-cfff-4b70-b994-3f9b19cce227:autoScalingGroupName/awseb-e-x2
dyyppidm-stack-AWSEBAutoScalingGroup-Q3gJgruA8DYe:policyName/awseb-e-x2dyyppidm-stack-AWSEBAutoScalingScaleDownPolicy-xOfrLZRHeOlw
2025-02-02 06:29:06    INFO    Created Auto Scaling group policy named: arn:aws:autoscaling:us-east-1:640168430080:scalingPolicy:1c8c2ed3-b156-4a97-86f1-b262b211742e:autoScalingGroupName/awseb-e-x2
dyyppidm-stack-AWSEBAutoScalingGroup-Q3gJgruA8DYe:policyName/awseb-e-x2dyyppidm-stack-AWSEBAutoScalingScaleUpPolicy-2m1xUa6OJmNV
2025-02-02 06:29:06    INFO    Created CloudWatch alarm named: awseb-e-x2dyyppidm-stack-AWSEBCloudwatchAlarmLow-GQw3tc29x6AJ
2025-02-02 06:29:06    INFO    Created CloudWatch alarm named: awseb-e-x2dyyppidm-stack-AWSEBCloudwatchAlarmHigh-Zagcln0VU2LK
2025-02-02 06:30:53    INFO    Created load balancer named: arn:aws:elasticloadbalancing:us-east-1:640168430080:loadbalancer/app/awseb--AWSEB-jz3DmDfnahsP/eb82d5a23d11b4b9
2025-02-02 06:30:55    INFO    Created Load Balancer listener named: arn:aws:elasticloadbalancing:us-east-1:640168430080:listener/app/awseb--AWSEB-jz3DmDfnahsP/eb82d5a23d11b4b9/29aaf48dc873d25b
2025-02-02 06:31:47    INFO    Instance deployment completed successfully.
2025-02-02 06:32:00    INFO    Application available at heart-tracking-app-env.eba-qzigkeuz.us-east-1.elasticbeanstalk.com.
2025-02-02 06:32:01    INFO    Successfully launched environment: heart-tracking-app-env

(Heart-Disease-App) (base) maxim-eyengue@Magic-Maxim:~/Heart-Disease-App$
```

maxim-eyengue@Magic-Maxim: ~/Heart-Disease-App

```
(ml-zoomcamp) maxim-eyengue@Magic-Maxim:~/Heart-Disease-App$ python predict_test_cloud.py
{'heart_disease': False, 'heart_disease_probability': 0.3}
The patient seems healthy: no treatment needed.
(ml-zoomcamp) maxim-eyengue@Magic-Maxim:~/Heart-Disease-App$
```


Maximilien Eyengue

Model Deployment



Maximilien Eyengue

Key Takeaways

What to do

- Be curious, judgmental and argumentative
- Think on how to improve

What not to do

- Rush into projects
- Skip data cleaning

There is always room for improvement…

?

Thanks!

Do you have any questions?

maximilien.boulou@aims-cameroon.org
+237 699 05 45 37

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**

Thanks also to the DataTalks.Club

Github Project Link