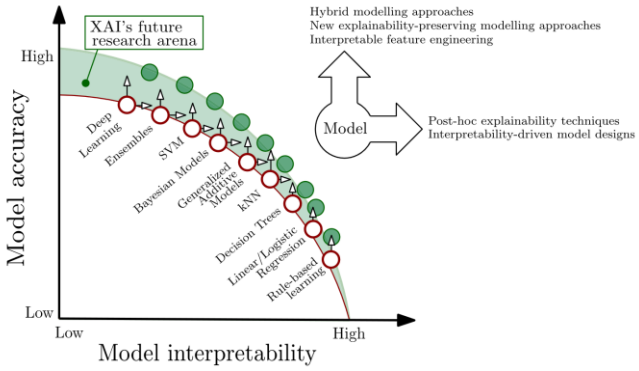
Exercise 5. eXplainable Artificial Intelligence in Healthcare

In this exercise, we will work with the same dataset that we used in exercise 4, i.e., the Pima Indians Diabetes Database[1]. Here, we will analyse the explainability of the different classifiers we used in the previous exercise and delve into the accuracy-explainability trade-off we have seen in the lectures, and as shown below:



For this practical exercise, the following libraries are needed to be installed. It is recommended to use the anaconda environment to install them, thus, open anaconda prompt program and run the following instructions:

- Eli5: `conda install eli5`
- Grpahviz: `conda install graphviz`

Note: If you are not using the anaconda environment, the above commands should be run in the folder where you have installed your python version and substitute the word '`conda`' by `pyhton.exe -m pip`

1. Explainability of transparent models

As we saw in exercise 4 the best classifier in terms of accuracy to classify patients in this dataset with or without diabetes was Logistic Regression. Logistic Regression is known as a transparent model which can provide an indication of the relevance that each of the variables has in the classification output.

Question 1.1. Provide the coefficient for each of the variables by using the built-in function `coef_` of Logistic Regression: sklearn.linear_model.LogisticRegression — scikit-learn 1.1.2 documentation. What is the variable that has the most influence on the positive classification of diabetes? And what is the one that has the least? How would you interpret the coefficients?

Note: You will need to create and train a classifier by using the Logistic Regresion in the same way we did in the Exercise 4.

Hint: To access to different steps of a pipeline (in our case the LogReg classifier) use the following syntaxis:

```
importance = clf_pipe['clf'].coef_[0]
```

To enhance the interpretation of the coefficients obtained, they could be plotted in a bar chart by running the following code:

[1] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, 'Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus', *Proc Annu Symp Comput Appl Med Care*, pp. 261–265, Nov. 1988

```
# plot feature importance
plt.bar([x for x in range(len(importance))], importance, tick_label=features)
plt.xticks(rotation=70)
plt.grid(axis='y', linestyle='--', )
plt.ylabel('LogReg coefficients value')
plt.xlabel('Features')
plt.show()
```

If you want to go into detail of the interpretation of Logistic Regression, we recommend you to review the slides of the lecture of this course 'DSS Method Development - 2' and visit this link: Logistic regression 1: from odds to probability | Dr. Yury Zablotski (yury-zablotski.netlify.app)

Question 1.2. In exercise 4 we use the Random Forest algorithm to build a classifier for predicting diabetes. The Random Forests algorithm belongs to the ensemble trees family where the algorithms employ different combinations of decision trees that act as base classifier to obtain an enhanced classification or regression performance. See the link for more information: 1.11. Ensemble methods — scikit-learn 1.1.2 documentation, and the slides of the Lecture 'DSS Method development 2'

Decision trees are well known for their transparency in their decision making based on the different conditions that construct a decision path that can be broken down into different rules. Therefore, by using the scikit-learn estimator `DecisionTreeClassifier` (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier) we can build a classifier similarly to the Random Forest one (for the sake of explainability avoid standardization this time) and calculate its accuracy, sensitivity and precision with the Test set. Set the parameters values similarly to the Random Forest, i.e: `max_depth=5, min_samples_split=8, min_samples_leaf=3, class_weight='balanced', random_state=42,`

Question 1.3 Decision Tree allows explaining its decision by a set of rules created about specific values of their features. In addition, the scikit-learn package has a method to depict the Tree (see https://scikit-learn.org/stable/modules/tree.html) in combination with the GraphViz library. Depict the Tree by using the following code and describe the set of rules for classifying an instance as diabetes as well as non-diabetes.

```
 from sklearn import tree
import graphviz
dot_data = tree.export_graphviz(tree_clf, feature_names=features,
class_names=tree_clf.classes_.astype(str),filled=True, rounded=True,special_characters=True)
#tree_clf is the name of the classifier created with DecisionTree
graph = graphviz.Source(dot_data)
graph.format = "png"
file_name_dt='Diabetes_Decision_Tree'
graph.render(file_name_dt)
```

2. Explainability of black-box models

Ensemble trees models, along with neural networks and deep learning models, are usually seen as black box models due to the difficulty of interpreting their decisions since they are based on complex logic within the algorithms they use. Thus, to improve the explainability of these models, data scientists have to use auxiliary techniques that are applied to the already developed models and are known as post-hoc techniques.

In this section, we will explore different post hoc XAI techniques that will allow us to analyze how the features influence the final classification output of a black box model. Thus, we will employ a Random Forest classifier similar to the one used in Exercise 4.

Create a Random Forest classifier and train it with the same train set we have used with the Logistic Regression. Initialize the Random Forest classifier with the following hyperparameters:

```
(max_depth=5, n_estimators=100, min_samples_split=8, min_samples_leaf=3, bootstrap= True, random_state=42)
```

a) Intrinsic feature importance

Since we have used Random Forest as the classifier estimator, scikit-learn library offers in its Random Forest implementation the possibility to know the importance of the different features by assessing the GINI (mean impurity decrease) when the trees are created (see attributes section in https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html).

Question2.1 Obtain the importance of the features and plot them. Which is the most important feature? Is it different from what we obtained with the Log Reg classifier?

Note: Remember that as we are using a pipeline, the access to the classifier estimator can be done with the following instruction:

```
clf_pipe['clf']
```

b) Feature permutation importance

Another explainability technique, that has been shown during XAI lesson is Feature permutation importance. Scikit-learn has a dedicated function to explore the change in the classifier performance by permuting their features' values.: sklearn.inspection.permutation_importance — scikit-learn 1.3.1 documentation.

Question2.2 To use this technique execute the following code and explain the figure generated in terms of y-axis values:

```
from sklearn.inspection import permutation_importance
import matplotlib.pyplot as plt
import numpy as np

# Compute permutation importance
result = permutation_importance(clf_pipe_RF, X_train, y_train, n_repeats=30, random_state=42)

# Sort features by importance
sorted_idx = result.importances_mean.argsort()

# Create a DataFrame to hold the results
feature_importance_df = pd.DataFrame({
    'Feature': np.array(features)[sorted_idx],  # Assuming 'features' contains the feature names
    'Importance_Mean': result.importances_mean[sorted_idx],
    'Importance_Std': result.importances_std[sorted_idx]
})

# Sort the DataFrame by Importance_Mean in descending order
sorted_feature_importance_df = feature_importance_df.sort_values(by='Importance_Mean',
ascending=False)
print(sorted_feature_importance_df)

# Plotting
plt.figure(figsize=(12, 8))
```

```
plt.barh(sorted_feature_importance_df['Feature'],
sorted_feature_importance_df['Importance_Mean'],
         xerr=sorted_feature_importance_df['Importance_Std'], align='center', alpha=0.5,
ecolor='black', capsize=10)
plt.xlabel('Importance', fontsize=20)
plt.ylabel('Feature', fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.title('Sorted Feature Permutation Importance', fontsize=20)
plt.gca().invert_yaxis()  # Reverse the order to have the most important feature at the top
plt.show()
```

c) Partial Dependence Plots

Partial Dependence Plots (PDP) is another XAI technique that is model-agnostic, i.e., it can be applied to any ML model. PDPs show the marginal effect of a given feature on the predicted outcome over the range of its observed values[2]. As a result, PDPs visualize the overall probability (y-axis) of a given model's output for every value of a certain feature, all the other feature values being equal.

Scikit Learn implements a function to show the PDP for a certain estimator: `from_estimator` method of `PartialDependenceDisplay` class (https://scikit-learn.org/stable/modules/generated/sklearn.inspection.PartialDependenceDisplay.html#sklearn.inspection.PartialDependenceDisplay.from_estimator).

Question2.3 By using the `from_estimator` method, depict the PDP of the 8 features and discuss the results obtained (i.e., probability trends for every feature). Note: to print the 8 PDPs in a bigger size it is recommended to use the following line code before calling the `from_estimator` method and set the four axes created (`ax1,ax2,ax3,ax4`) as a parameter in the method call:

```
from sklearn.inspection import PartialDependenceDisplay

fig_train, (ax1,ax2,ax3,ax4) = plt.subplots(4, 2, figsize=(20, 12))
PartialDependenceDisplay.from_estimator(clf_pipe, X_train,X_train.columns, ax=[ax1,ax2,ax3,ax4])
```

3. Trustworthy assessment of the diabetes prediction model

During the lesson on XAI, the concept of trustworthy AI was discussed, along with the set of requirements and sub-requirements that an AI system must meet to be considered trustworthy. In a nutshell, the principles and requirements are outlined in the table below:

Requirements	Sub-Requirements
Human Agency and oversight	Fundamental rights, human agency and human oversight.
Technical Robustness and Safety	Resilience to attack and security, fall back plan and general safety, accuracy, reliability and reproducibility
Privacy and data governance	Respect for privacy, quality and integrity of data, and access to data
Transparency	Traceability, explainability and communication
Diversity, non-discrimination and fairness	Avoidance of unfair bias, accessibility and universal design, and stakeholder participation
Societal and environmental wellbeing	Sustainability and environmental friendliness, social impact, society and democracy
Accountability	Auditability, minimisation and reporting of negative impact, trade-offs and redress.

[2] J. H. Friedman, 'Greedy Function Approximation: A Gradient Boosting Machine', *Annals of Statistics*, vol. 29, pp. 1189–1232, 2000.

For more information, refer to the following document, which has also been uploaded to Moodle: Ethics guidelines for trustworthy AI | Shaping Europe's digital future (europa.eu)

In this section, you must elaborate on how you would apply the ideas derived from Trustworthy AI to the Diabetes prediction model we have been working on. To do so, it is essential to understand the context in which the prediction model will be deployed. Consider the following use case:

You are a member of a decision-making committee that wants to deploy a diabetes screening service based on a Computer-Assisted Diagnosis (CAD) system for use in rural areas of a European country. Conventional methods for diagnosing diabetes are impractical in these regions due to the absence of specialized healthcare facilities. As such, the CAD system offers significant advantages by enabling early detection of potential diabetes cases, which can then be referred to regional hospitals for more detailed follow-up. The management board has requested that the CAD system should comply with Trustworthy AI principles.

This CAD system will employ the diabetes prediction model we have been working on in this exercise. The screening service will be performed by general practitioners in the healthcare centers of the areas. The workflow of the screening services is as follows:

- *The GP receives the patient in the consultation room and collects all the necessary data for the prediction model.*
- *The GP inputs the features' values into the system, and the GP obtains only the output result of the model; whether the patient is likely to have diabetes or not.*
- *In the event of a positive diagnosis, the patient is referred to the hospital region for further evaluation and treatment.*

Question 3.1

-Select two sub-requirements from the table above and describe (1 paragraph of 4-5 lines per each sub-requirement) how would you address them in the CAD system described above. You may propose new functionalities for the system, collect new data, modify the training or testing phase, or interact with other actors/stakeholders.

Question 3.2

- Could you identify any tension or trade-off between the different sub-requirements in this system? Please, elaborate your answer (1 paragraph of 4-5 lines)