Final Report

Brayden Edwards

AI 541 - Oregon State University

Final Project Report Draft

Spring Term - June 10, 2025

1. Problem To Be Solved

The Problem

The prediction problem of my final project is the classification of patients as either not having diabetes or being prediabetic/diabetic. While the goal is primarily based on predicting these two classes, I also aim to create separate models for two distinct user groups: low-context at-home users and domain experts.

Users

Because the use case of these groups are vastly different, I have implemented different output metrics for each model. For instance, the domain expert model provides an estimated probability the test example is positive and a class prediction, while the at-home model only provides binary class prediction. Additionally, both of these models utilize different features with the goal of increasing the usability of the at-home model by lowering the barrier for usage. This is done by removing the `HighChol` feature as it cannot be reasonably assessed by an at-home user.

The goal of the expert model is to work alongside clinicians rather than replacing them; I don't want my model to act as a sole decision-maker, but rather as a decision support tool—offering input for diabetes risk assessment. To do this, I hope to create a model that mitigates automation bias. Automation bias occurs when a model repeatedly makes correct decisions, leading the user to simply trust the model in the future. My hope is that the focus on probability outputs will work to counteract this tendency.

Nevertheless, I will not be avoiding a threshold decision. Physicians may suffer from decision fatigue, meaning if a model output is too complex or lacks interpretability its use may be avoided. While my focus is on calibrated probability outputs for the domain expert model, I have explored different threshold methods to ensure that the decision maximizes benefit to the domain expert. The at-home model, on the other hand, has been configured to provide the user with a warning if they are deemed to be at risk for diabetes. This warning is intended to prompt users to seek a physical assessment by a physician.

2. Dataset Properties

Data Source

The datasource I utilized is the **CDC Diabetes Health Indicators,** authored by the Centers for Disease Control and Prevention (CDC)**.** This dataset was found through the UC Irvine Machine Learning Repository. Here is a link to the dataset through the UC Irvine Machine Learning Repository website:

https://archive.ics.uci.edu/dataset/891/cdc+diabetes+health+indicators

Dataset Profile

The table below defines a global view of the dataset, showing various important dataset metrics:

Number of items	Number of classes	Class distribution
253680	2	Neg: 86.1% — Pos: 13.9%

Below is a list of all features and their type, either numerical or categorical.

Feature	Type
HighBP	categorical
HighChol	categorical
CholCheck	categorical
BMI	numeric
Smoker	categorical
HeartDiseaseorAttack	categorical
PhysActivity	categorical
Fruits	categorical
Veggies	categorical
HvyAlcoholConsump	categorical
AnyHealthcare	categorical
NoDocbcCost	categorical
GenHlth	numeric
MentHlth	numeric
PhysHlth	numeric
DiffWalk	categorical
Sex	categorical
Age	numeric
Education	numeric
Income	numeric

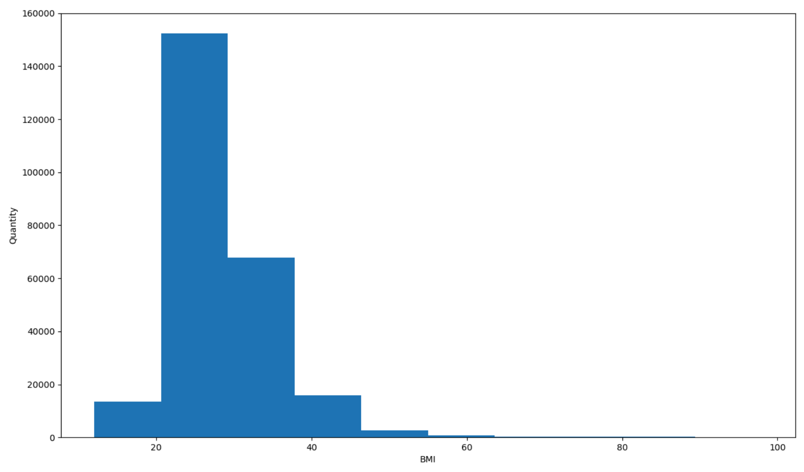
Numeric Features

Below is a table providing information for all the numeric features in my dataset.

Feature	Min	Max	Mean	Median	Number Missing
BMI	12	98	28.3824	27	0
MentHlth	0	30	3.1848	0	0
PhysHlth	0	30	4.2420	0	0
Age	1	13	8.0321	8	0
Education	1	6	5.0504	5	0
Income	1	8	6.0539	7	0

Interestingly, we see that the minimum value in both `MentHlth` and `PhysHlth` are 0, yet the UC Irvine Machine Learning Repository website states that the range of values for these two features are 1-30. However, given that these features represent the number of days during the past 30 days that the patients mental and physical health were not good, it makes sense that some people would have 0 days which they struggled either mentally or physically.

It’s also worth noting that BMI has a wide and extreme range of 12, meaning a very low BMI, and 98 meaning a high one. Below is a chart to help visualize the spread and the lack of representation in the upper end:



Although there is a wide spread, the average BMI in the U.S. is 29.27 [U.S. Department of Agriculture], meaning that my data is fairly representative of the population as the mean BMI of my dataset is 28.3824.

Categorical Features

Below is a table providing information for all the categorical features in my dataset

Feature	Distinct Values	Number Missing	Possible Values
HighBP	2	0	0 or 1
HighChol	2	0	0 or 1
CholCheck	2	0	0 or 1
Smoker	2	0	0 or 1
HeartDiseaseorAttack	2	0	0 or 1
PhysActivity	2	0	0 or 1
Fruits	2	0	0 or 1
Veggies	2	0	0 or 1
HvyAlcoholConsump	2	0	0 or 1
AnyHealthcare	2	0	0 or 1
NoDocbcCost	2	0	0 or 1
DiffWalk	2	0	0 or 1
Sex	2	0	0 or 1

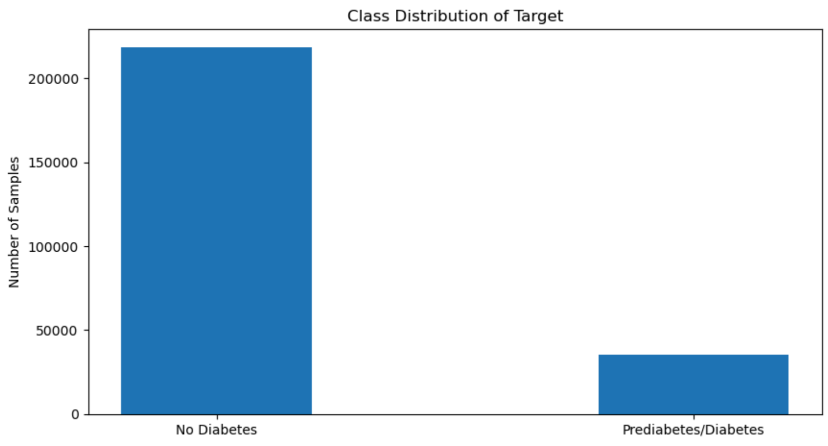
Target Column

The target column for this dataset is `Diabetes_binary`, which is a binary feature consisting of either 0 or 1, where:

- 0 = no diabetes
- 1 = prediabetes or diabetes

The class distribution is shown below via table and bar chart:

Value	Number of Value	Percent of Value
0	218334	86.1%
1	35346	13.9%



Interestingly, my dataset actually under-represents the positive class when looking at the true distribution in the United States. According to the CDC, 11.6% of the U.S. population are diabetic. The CDC also states that the crude, less accurate methods puts percentage of prediabetic adults at 38.0% while a stricter, clinical-level estimate is around 10.8% [Centers for Disease Control and Prevention]. Therefore, a more appropriate class balance should be around 23.4% for the positive class by combining both diabetic and clinical-level-certainty prediabetic estimates.

Data Examples

Below are three examples from my dataset. Please note that these examples have been transposed to ensure readability. Thus, the first column is the header column and the remaining columns are examples; the rows are then features.

Feature	Example 1	Example 2	Example 3
HighBP	1	0	1
HighChol	1	0	1
CholCheck	1	0	1
BMI	40	25	28
Smoker	1	1	0
Stroke	0	0	0
HeartDiseaseorAttack	0	0	0
PhysActivity	0	1	0
Fruits	0	0	1
Veggies	1	0	0
HvyAlcoholConsump	0	0	0
AnyHealthcare	1	0	1
NoDocbcCost	0	1	1
GenHlth	5	3	5
MentHlth	18	0	30
PhysHlth	15	0	30
DiffWalk	1	0	1
Sex	0	0	0
Age	9	7	9
Education	4	6	4
Income	3	1	8
Diabetes_binary	0	0	0

3. Machine Learning Models

For this project, I used a Random Forest and XGBoost classifier models for each user group—four distinct models in total. I chose the Random Forest because I am familiar with the architecture and it allows for simple changes in class weights to assist with the class imbalance. The XGBoost model is something I am less familiar with but wish to compare results. Both models are tree based meaning they have some inherent interpretability, which is important for a healthcare domain.

The at-home model and the expert model required separate model instances because they differ in features. The expert model trains and evaluates on all the features in the dataset. In contrast, the at-home model trains on the same dataset without the `HighChol` feature. This feature is difficult to assess for the at-home user, so it was removed.

Both models share similar weaknesses, namely, they are computationally slow and have less interpretability than a linear model. The slow computation issue was especially apparent when performing a grid search for the best hyperparameters. To overcome this, I used OSU's HPC cluster. To address the issue of interpretability, I have deployed a few model agnostic post-hoc methods such as Lime, SHAP, counterfactuals, and Permutation Feature Importance.

Hyperparameter Tuning

To choose the best hyperparameters, I wrote a script that performed a grid search across a multitude of hyperparameters and found the combination that most maximized the F1 score. Here is the param list that the grid search used:

```
param_rf = {
    "n_estimators": [50, 100, 150],
    "max_depth": [None, 4, 10, 20],
    "class_weight": [
        "balanced",
        {0: 1, 1: 4},
        {0: 1, 1: 2},
        {0: 1, 1:6}
    ]
}

params_xgb = {
    'min_child_weight': [1, 5, 10],
    'gamma': [0.5, 1, 1.5, 2, 5],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'max_depth': [3, 4, 5],
    "scale_pos_weight": [4, 6, 8]
}
```

For the domain expert models, the grid search found the following parameters to be those that most maximized F1 score.

```
rf_params = {
    'class_weight': {0: 1, 1: 4},
    'max_depth': 10,
    'n_estimators': 100
}

xgb_params = {
  'colsample_bytree': 0.6,
  'gamma': 1,
  'max_depth': 3,
  'min_child_weight': 1,
  'scale_pos_weight': 4,
  'subsample': 1.0
}
```

For the at-home models, the grid search found the following parameters to be those that most maximized F1 score.

```
rf_params = {
    'class_weight': {0: 1, 1: 4},
    'max_depth': 10,
    'n_estimators': 150
}

xgb_params = {
    'colsample_bytree': 1.0,
    'gamma': 5,
    'max_depth': 3,
    'min_child_weight': 10,
    'scale_pos_weight': 4,
    'subsample': 1.0
}
```

To ensure that no target leaking occurred, I used the same train, validation, test split throughout the entire project. This was done by declaring and using test size, validation size, and random seed variables. By doing this, I ensured that the same test set was stripped from all forms of training and only used for model evaluation.

4. Evaluation

I am using a variety of metrics for evaluating my model's performance, such as:

- F1 Score - balances precision and recall, which will prove useful for the class imbalance prevalent in the dataset.
- Recall - the true positive rate will provide insights into the model's performance by identifying how well the model determines actual diabetic/prediabetic patients.
- Precision - provide the fraction of patients classified as diabetic/prediabetic that were actually diabetic/prediabetic.
- Balanced Accuracy - provides a class-imbalanced-aware metric for overall model performance. This was computed using Scikit-Learn built in metric [11].

Using a combination of metrics will provide a comprehensive measure of the model's performance.

I employed nested k-fold cross-validation for hyperparameter tuning. The inner loop was used to identify optimal hyperparameters using a grid search, while the outer loop provided unbiased evaluation of those choices. The nested k-fold cross-validation was performed only on the training set.

The hyperparameter tuning process was computationally expensive and was only performed once. After my initial script found the best performing configuration of parameters, I noted the best parameters and used these for the remainder of the project. The model was then refit to the same training set, while leaving the validation and testing set untouched. The training/validation/testing data was split as 60%/20%/20% of the total respectively with the same random state to ensure no leakage. Specifically:

- Training set: used initially to find the best hyperparameters and then used in other scripts to fit the model with the chosen hyperparameters
- Validation set: used to calibrate the model's predicted probabilities
- Test set: used only once at the end to assess the model's generalization performance

Accuracy is not a useful metric because of the major class imbalance in my dataset. As a result, defining a baseline is tricky. The key baseline I am considering when looking at my results is how well the model is identifying at risk patients while balancing the cost of false positives. I also deem any model that cannot achieve a recall on the positive class above the class distribution 13.9% to be useless.

Model Calibration

The calibrated probability outputs were found by calibrating my models through Scikit-Learn's calibration library. Specifically, I used the CalibratedClassifierCV function with the isotonic method. I used the isotonic method because it generally performs as well or better as the sigmoid method when the dataset is large enough, according to Scikit-Learn's documentation [12]. I did experiment with both and found the isotonic method to be slightly better. It's also important to restate that calibration happened on the validation set and was the only use case for the validation set, as nested-cross validation was done on the training data during hyperparameter tuning. For the isotonic method in particular, model calibration is accomplished through minimizing squared error loss between predicted probabilities and the true label [12].

Once the model's probability outputs were computed, I found the threshold that most maximized F1 score. All calibrated probability outputs greater than that threshold were then predicted as positive; otherwise, they were predicted negative.

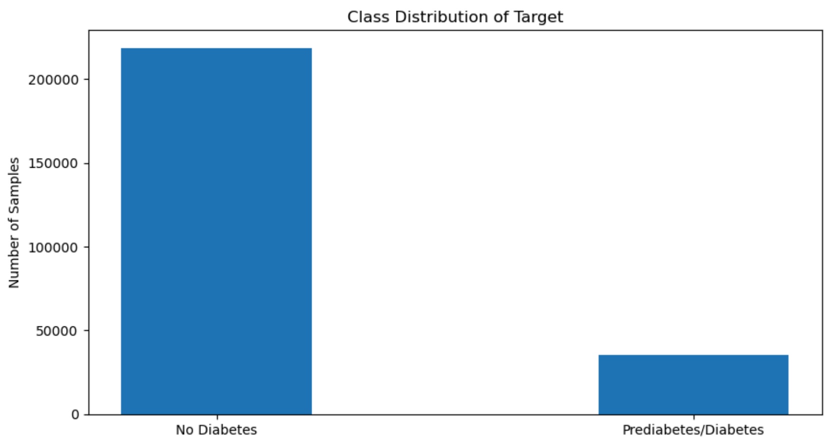
5. Challenges

Here are some key challenges that will be addressed in my final project.

Challenge 1: Class imbalance

The target classes in my dataset are highly imbalanced, significantly favoring no diabetes. This is to be expected as not having diabetes is more prevalent than being prediabetic/diabetic. Here is a bar chart displaying the level of class imbalance that exists:

Value	Number of Value	Percent of Value
No Diabetes	218334	86.1%
Prediabetes/Diabetes	35346	13.9%



To address this class imbalance, I plan to use the following three methods.

Do Nothing

First, I have done nothing to address the class imbalance. This means that my model will have the following parameters:

domain expert models

```
rf_params = {
    'max_depth': 10,
    'n_estimators': 100
}

xgb_params = {
  'colsample_bytree': 0.6,
  'gamma': 1,
  'max_depth': 3,
  'min_child_weight': 1,
  'subsample': 1.0
}
```

At home models

```
rf_params = {
    'max_depth': 10,
    'n_estimators': 150
}

xgb_params = {
    'colsample_bytree': 1.0,
    'gamma': 5,
    'max_depth': 3,
    'min_child_weight': 10,
    'subsample': 1.0
}
```

This model will act as my baseline for comparison regarding class imbalance. F1 score, recall, precision, and balanced accuracy will be provided for model comparison.

Resampling

Another method for addressing class imbalance is undersampling the majority class. I used the random under sampler from the Imbalanced Learn library with `sampling_strategy='auto'` , which randomly removes samples from the majority class until the classes are balanced. I tested this approach using the same model parameters as in the 'Do Nothing' baseline.

Synthetic Data Generation

I also used synthetic data to see how my model performs. To do this, I utilized SMOTENC from the Imbalanced Learn library [13]. SMOTENC is a SMOTE technique that is compatible for generating synthetic data from a dataset with both numeric and categorical features. I configured SMOTENC with `sampling_strategy='auto'` , which oversamples the minority class to match the majority class size ensuring a fully balanced dataset. During testing, I used the parameters described in the "Do Nothing" section above. It is also important to note that SMOTENC was fitted only on the training data; the testing data remained untouched.

Cost Sensitive Learning

Finally, I used cost sensitive learning by penalizing my model for misclassifying the minority class. Both machine learning models I used, Random Forest and XGBoost, support class weighting to help address class imbalance. The parameters I used for these models are those described in the "Machine Learning Models" section. Specifically, the Random Forest had `'class_weight': {0: 1, 1: 4}` and the XGBoost model had `'scale_pos_weight': 4` .

Baseline

For comparison, I have also established a baseline prediction of the majority class. This will allow us to compare results with the baseline to ensure that the models perform better.

Challenge 2: Interpretability

The desire for interpretable models in the medical domain is ever growing. Those who use models to make predictions often wish to understand how the model makes its decisions. Interpretability is also important for responsible machine learning practitioners who wish to understand how and why their model is making its decisions.

In my final project, I tackle the issue of interpretability. As my dataset exists in the medical domain and interpretability is vital to real world machine learning, I feel that this is an important challenge to address. I plan to do the following to enhance interpretability.

Global Model Agnostic Post-hoc Interpretability

For global methods, I intend on using Permutation Feature Importance (PFI) and global SHAP values to assess the overall impact of features on the models predictions.

Permutation Feature Importance (PFI) is a model-agnostic technique used to estimate the importance of each feature by measuring the change in model performance when that feature's values are randomly shuffled. The idea is that if a feature is important, changing its relationship with the target will hurt the model's performance. PFI provides an intuitive and interpretable way to assess which features influences the model's predictions on a global level. For this challenges, I utilized Scikit-Learn's built in permutation feature importance method with F1 scoring and 5 repeats [6].

SHAP (SHapley Additive exPlanations) is a method for global and local model interpretability based on Shapley values, introduced by Lloyd Shapley in 1953, which are rooted in cooperative game theory [7]. The basic idea is to assign features a certain value based on their contribution in predicting the target. Put simply, this is done by sampling a subset of features, predicting on the target, introducing a new feature to the set, re-predicting on the target, and assessing the change in prediction with the new feature introduced. This process is repeated multiple times, over many feature combinations, and the average change in prediction for each feature is computed. These changes per feature are then sorted, with the most important features being those that created the most change. SHAP is most fundamentally a local explanation method as it explains individual predictions. However, by taking the average SHAP values across many instances, SHAP can easily be turned into a global method for explaining feature importance. Along with that, SHAP also has the ability to describe how each feature's values contributed to a specific output. For example, if an increase in BMI causes the model to predict the positive class more often, SHAP can reveal that model behavior [2].

Local Model Agnostic Post-hoc Interpretability

I plan on using methods such as LIME [10] and SHAP [8] to assess how each feature contributed to a specific prediction on the local level.

LIME (Local Interpretable Model-Agnostic Explanations) is a model agnostic technique for interpreting individual predictions by approximating it locally with an interpretable, simpler model. This is usually done with a linear model. It works by perturbing the input data around a specific instance and observing how the model's predictions change. LIME perturbs the input around a target instance, fits a linear regression line to a neighborhood, and identifies which features most influenced the prediction. This results in an interpretable local explanation for why the model made its decision for that input. In my case, I used the LIME Python library to perform this challenge [10].

I also used SHAP for local methods to assess interpretability. As previously stated, SHAP is primarily a local explanation method, as it provides key insights into why the model made a certain prediction. The local SHAP method works similarly to the global version described above, but instead of explaining a subsample of data, it explains a single instance.

Provide Contrastive Explanations

I also intend on assessing counterfactuals; this will allow me to provide information about how certain metric changes could lead to a different classification. My goal is to provide this information to a domain expert or an at-home user to make them aware of what to avoid or improve upon.

Challenge 3: Representation

In my final challenge, I will address representation of features. The motivation behind this challenge is to decrease barrier for obtaining a prediction. This is particularly crucial for the at-home model, where users may have time constraints or feel hesitant or uncertain answering the questions needed to make the prediction. Removing features lowers the mental threshold of obtaining a prediction for the at-home user. To assess model performance between each iteration of feature removal, I used F1, Recall, Precision, Balanced Accuracy, and Brier score. Each iteration has it's own trade-offs, which is discussed further in my results.

Do Nothing (All Features)

In the simplest form, I plan on doing nothing. In other words, I will begin by training on all the features available. This will be especially useful for my domain expert model, but I also plan to assess how performance drops as I remove features.

Remove `HighChol` Feature

As previously discussed, I will remove the `HighChol` feature from my dataset. Primarily, I am doing this to ensure that an at-home user has full access to the model. After looking at all the features present in the dataset, `HighChol` is the only one that doesn't appear assessable at home. Therefore, by removing this feature, I will be removing the barrier for use.

General Feature Removal

In the spirit of further removing barriers for use, I have performed number of tests where I remove certain features that require additional effort from the user beyond what they simply know. For example, I looked at removing two different categories of features:

- **Calculation Features:** these are features that require the user to perform some type of conversion. There are three of these features that I chose to remove: `Income`, `Education`, & `Age`.
    - `Income`: this feature requires the user to convert their income level into a scale from 1 to 8 as defined in the dataset's codebook with 1 being less than $10,000, 5 being less than $35,000, and 8 being $75,000 or more, for example.
    - `Education`: this feature requires the user to input their education level as described in the dataset's codebook. This feature is on a scale of 1-6 where 1 is never attended school or only kindergarten, etc.
    - `Age`: this feature requires the user to input their age as described in the dataset's codebook. This features is on a scale of 1-13, with 1 being 18-24 years of age, 9 being 60-64 years of age, and 13 being 80 or older, for example.

  These types of features create an unwanted barrier for potential users, by requiring them to perform extra work. I did consider removing BMI, as it does require a form of calculation, but it was too informative for my models—removing it significantly dropped the ability of my models to assess for diabetes. Also, the calculation method for BMI is not dataset specific unlike the three features above, which have arbitrary conversions based on the CDC's decisions at data collection time.

- **Health Survey Features:** these are features that require the user to provide general assessments of health. This includes the following features: `MentHlth`, `PhysHlth`, & `GenHlth`.
    - `MentHlth`: this features asks the user to provide the number of days of poor mental health over the past 30 days.
    - `PhysHlth`: this feature asks the user to provide the number of days that they experienced physical illness or injury in the past 30 days.
    - `GenHlth`: this asks the user, quite vaguely, to assess their general health on a scale of 1-5 with 1 being excellent, 2 very good, 3 being good, 4 fair, and 5 poor.

The features describe above are in contrast with more simple input features like `Sex` or `Smoker`, which simply require the user to input a simple binary answer without further conversion or consideration. The hope is that providing the user with an option to not answer the more difficult questions, such as the six features describe above, will yield more at-home users by decreasing the barrier for use.

6. Results

All results were obtained using random state 42. All my code utilizes a `RANDOM_STATE` variable in the `config.py` file to ensure reproducibility.

Challenge 1 Results: Class imbalance

Random Forest Model

Method	F1 Score	Recall	Precision	Balanced Accuracy	Num. of Train Samples
Do Nothing	0.467	0.603	0.381	0.722	162355
Undersample	0.397	0.899	0.255	0.737	45243
SMOTENC	0.411	0.801	0.277	0.731	279467
Cost Sensitive Learning	0.462	0.543	0.402	0.706	162355

XGBoost Model

Method	F1 Score	Recall	Precision	Balanced Accuracy	Num. of Train Samples
Do Nothing	0.471	0.607	0.384	0.725	162355
Undersample	0.407	0.887	0.264	0.743	45243
SMOTENC	0.401	0.772	0.271	0.718	279467
Cost Sensitive Learning	0.47	0.606	0.384	0.724	162355

Baseline Model (Majority Class Classifier)

Method	F1 Score	Recall	Precision	Balanced Accuracy
Do Nothing	0.0	0.0	0	0.5
Undersample	0.245	1.0	0	0.5
SMOTENC	0.245	1.0	0	0.5
Cost Sensitive Learning	0.141	0.141	0	0.501

For this challenge, cost sensitive learning produced the best results. A close second was the baseline where my strategy was to do nothing. This method produced surprisingly good results given the class imbalance in my dataset. Undersampling and SMOTENC, on the other hand, yielded weaker results, especially in terms of F1 score.

It's worth noting that while the Do Nothing and Cost Sensitive Learning strategies largely obtained the same results, I chose to favor Cost Sensitive Learning as both methods were tested during my nested k-fold cross validation hyperparameter tuning, and the class weights used in the Cost Sensitive Learning method proved to obtain better generalization during the cross validation. Therefore, while they obtain basically the same results on the hold-out set, Cost Sensitive Learning likely generalizes better to new data.

The resulting lower F1 for SMOTENC align with findings in the literature. Elor and Averbuch-Elor (2022) found that when models use thresholding on calibrated probabilities to maximize F1—exactly my approach with Random Forest and XGBoost—SMOTE methods tend to underperform on expressive models [1]. They also observed that balancing did not improve AUC/ROC for tree-based classifiers, which is consistent with my results. All outcomes performed better than the baseline classifier of predicting the majority class, indicating that my model is sufficiently learning from the data and generalizing to unseen data.

Challenge 2 Results: Interpretability

Global Model Agnostic Post-hoc Interpretability

Permutation Feature Importance

Below is the output of my Permutation Feature Importance test. This test was performed across three random samples. For reproducibility, those samples have been labeled with the random seed associated with the sampling method.

Every column where there was not absolute agreement in mean importance across all three feature importance output is highlighted in blue.

Feature	Mean Importance **Random Seed = 0**	Mean Importance **Random Seed = 10**	Mean Importance **Random Seed = 42**
GenHlth	0.068	0.068	0.068
HighBP	0.056	0.057	0.057
BMI	0.056	0.056	0.055
Age	0.028	0.028	0.028
HighChol	0.024	0.024	0.024
HvyAlchoholConsump	0.006	0.006	0.005
Income	0.005	0.005	0.005
HeartDiseaseorAttack	0.005	0.005	0.004
CholCheck	0.004	0.004	0.004
DiffWalk	0.004	0.004	0.001
Sex	0.001	0.001	0
Education	0.001	0.001	0
AnyHealthcare	0	0	0
Stroke	0	0	0
Veggies	0	0	0
PhysHlth	0	0	0
Smoker	0	0	0
Fruits	0	0	0
PhysActivity	0	0	0
NoDocbcCost	-0.001	0	-0.001
MentHlth	-0.001	-0.001	-0.001

The results across the three different feature importance outputs are remarkably similar given the property that feature importance results are known to vary greatly across different random states [2]. This is likely due to using thirty fit and test repeats (using the ‘n_repeats’ parameter in the ‘permutation_importance’ method) instead of Scikit-Learn’s default value of five [6]. When I used the default value of five, there were more inconsistencies across the three outputs, even resulting in the order of variables changing slightly. While using thirty repeats is more computationally expensive, it is worth the computation/time trade-off as the results are much more stable for only a small trade in computation time. Also, permutation feature importance only needs to be computed once, so an increase of 30 seconds to 3 minutes for better outputs is worth it.

Global SHAP Feature Importance

Methodology

For this interpretation method, I utilized SHAP values to gain insight into feature importance and each feature’s direction effect on model the models prediction. To implement this, I utilized the SHAP python library [8].

I utilized the same training, validation, and testing split across the entire project. I used the Random Forest and XGBoost model on all the features. The model parameters used on the Random Forest and XGBoost are as follows:

```
rf_params = {'class_weight': {0: 1, 1: 4}, 'max_depth': 10, 'n_estimators': 150}

xgb_params = {'colsample_bytree': 0.6, 'gamma': 2, 'max_depth': 5, 'min_child_weight': 1, 'scale_pos_weight': 4, 's
ubsample': 1.0}
```

A random, stratified sample was taken of 1000 data points from the testing set. The data was stratified by the target in the test set. This was done to ensure the same data distribution across in the random sample as the entire testing dataset. 1000 data points was chosen relatively arbitrarily. I started with the entire testing dataset, but was unable to produce SHAP values in a reasonable amount of time. I then decreased to a third, a sixth, 4000, and then 1000, adjusting based on time needed to run the SHAP values. 1000 was the first sample size that produced SHAP values in a timely manner. The approach of using a smaller sample size for estimation Shapely values to reduce computation time is supported by Molnar in chapter 17, with the caveat of increased variance in the estimate [2].

I then declared the SHAP 'Explainer' method, found in the docs here [8]. For the model input, I passed the custom prediction function in my class for both the Random Forest and XGBoost models, along with the stratified, random sample of 1000 datapoints. I also set the seed variable in the 'Explainer' method to 42. All the other variables in the 'Explainer' method were left as default, most notably the 'algorithm' variable which set to 'auto' by default, which "attempts to make the best choice given the passed model and masker [8]. In this case, the 'Explainer' selected the 'permutation' algorithm.

Global SHAP Results

The results from the SHAP analysis results were exceedingly interpretable and insightful for both the Random Forest and XGBoost model. There was high consistency across both models in what was considered the most important features, but there were real differences in how these features affected the models predictions. In this section, we will first look at the SHAP output of the Random Forest and then move onto the XGBoost model for analysis and comparison.

Below is the beeswarm plot showing my Random Forests SHAP outputs for the 1000 samples. Each feature appears along the y-axis in order of importance, with higher-up features contributing more to the model's prediction on average.

Each dot represents a single prediction for one person in the dataset. The position of the dot along the x-axis shows how much that features pushed the model's prediction away from the average output—either increasing it to the right towards a positive prediction or decreasing it to the left toward a negative one.

Color indicates the feature's value for that person: red for high values and blue for low ones. This makes it easy to spot trends. For example, if red dots consistently appear to the right of center, then higher values for that feature tend to increase prediction of the positive class.

With that, here is the beeswarm plot for the Random Forest's SHAP values:

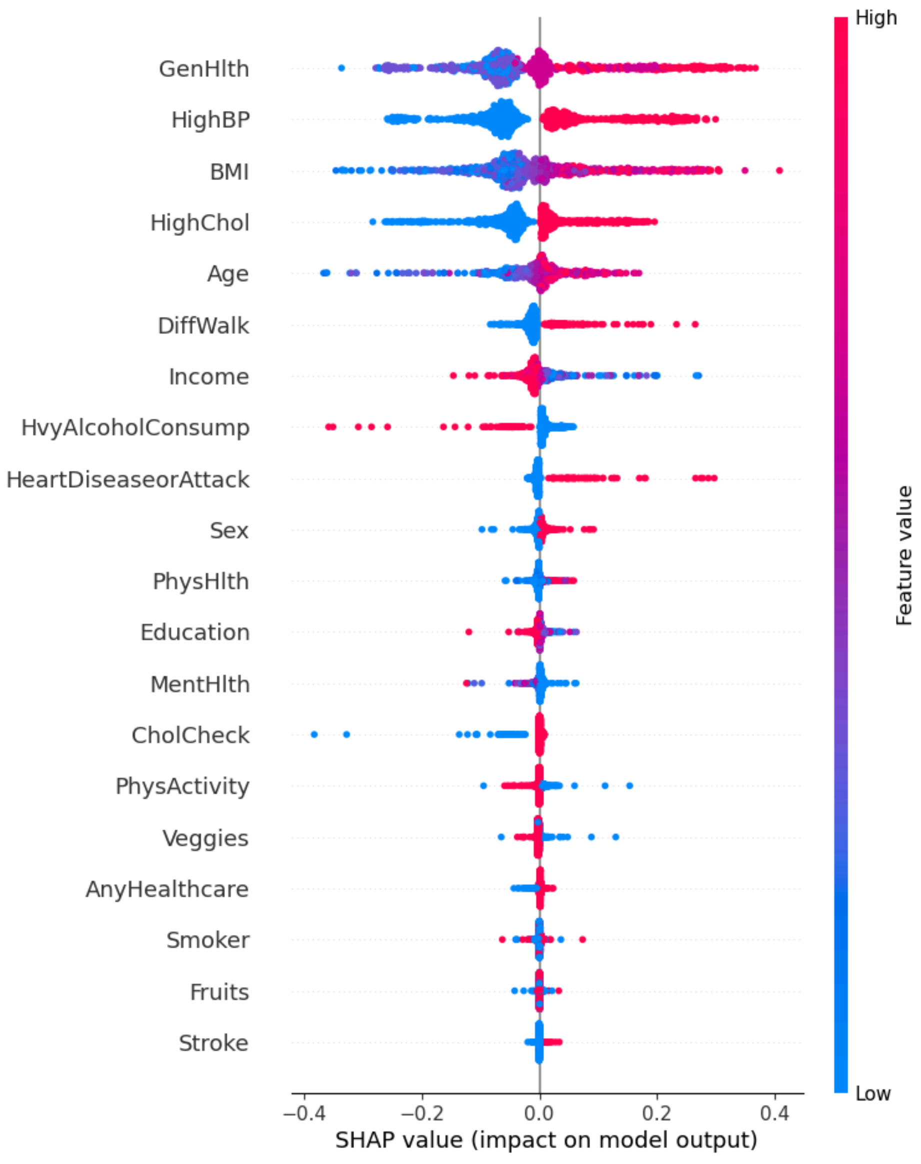


Figure: SHAP Beeswarm Plot for Random Forest Model (n=1000)

For the Random Forest model, we see that `GenHlth`, `HighBP`, and `BMI` the most important features for making predictions and follow the typical trends we would expect.

- Poor general health (high values in `GenHlth`): when general health is low, there is a strong push towards predicting the positive class.
- High blood pressure (`HighBP` = 1): individuals with high blood pressure are almost certainly more likely to belong to the positive class.
- Higher BMI: individuals with a high BMI are strongly associated with increase model output, meaning they are more likely to be prediabetic/diabetic.
- High Cholesterol (`HighChol` = 1): those with a high cholesterol are considered more likely to cause the model to predict the positive class.

There were a handful of features that seemed particularly noteworthy:

- Age: older individuals were more at risk of being labeled prediabetic/diabetic.

- Difficulty walking: individuals who stated they had serious difficulty walking or climbing stairs strongly associated with the positive class.
- Income: those who had a higher income were associated with a lower risk of being prediabetic/diabetic. This feature is particularly interesting as income is a feature that correlates with the target instead of being a casual one, yet there appears to be consistent model behavior in how this feature affects prediction outcomes.
- Sex: the model predicts that men are more likely to produce a positive outcome.

Now we move on to look at the XGBoost model's SHAP beeswarm plot on the 1000 samples for analysis:

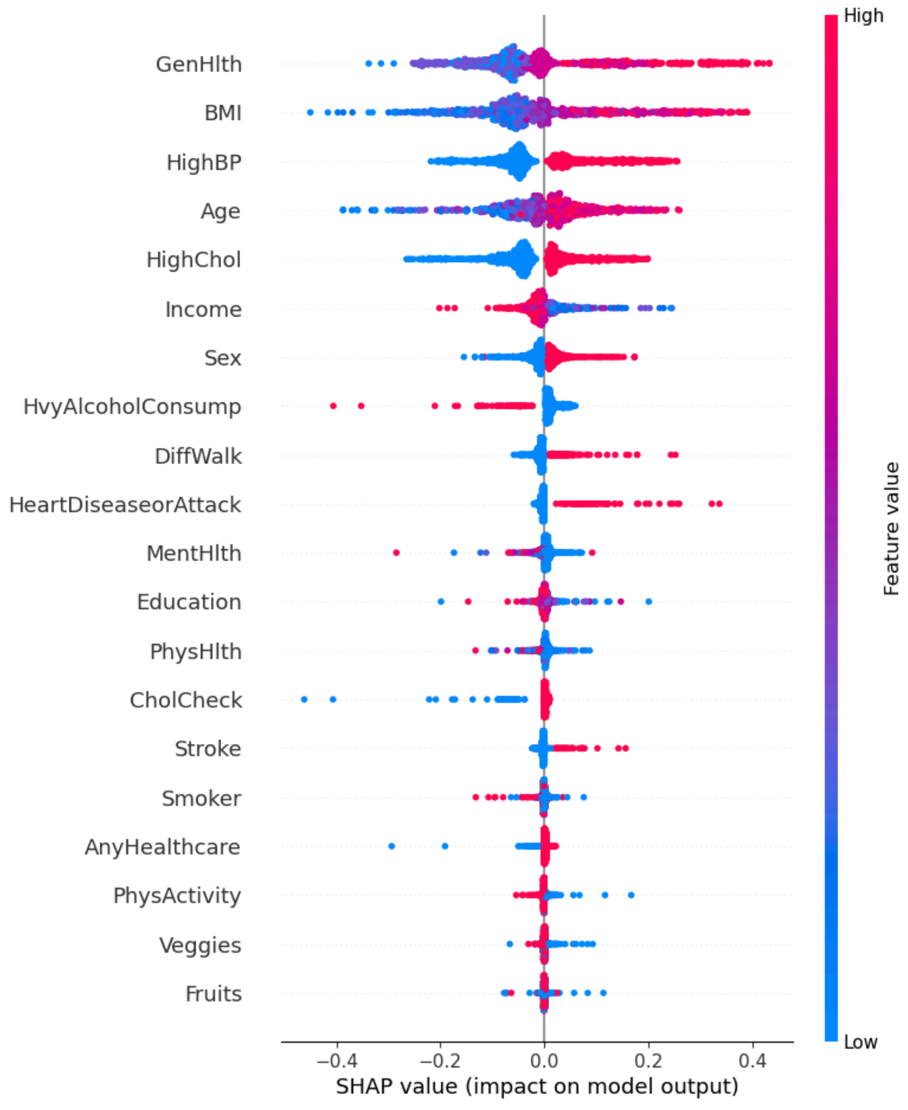


Figure: SHAP Beeswarm Plot for XGBoost Model (n=1000)

The resulting XGBoost model's SHAP outputs were similar to the Random Forest with some changes in ordering of features:

- BMI was considered a more important feature than high blood pressure, which is opposite of the Random Forest model.
- Difficulty walking dropped in importance from sixth (in the Random Forest model) to ninth place.
- The sex feature increased in important with even more polarization in the XGBoost model.

- The stroke indicator variable became more important in the XGBoost model's predictions compared with the Random Forest and became significantly more polarized.

We also see that the XGBoost model's SHAP values have a larger spread across nearly all variables indicating possible increased confidence in the XGBoost model. Finally, the `HighChol` feature appeared to be slightly more important in the Random Forest model, possibly providing implications for which model is better for the at-home user.

Comparing Permutation and Global SHAP Feature Importance

While both SHAP and permutation feature importance (PFI) seek to quantify the contribution of each feature to the model's prediction, they do so from different angles. Yet despite these methodological differences, the results between PFI and SHAP were surprisingly consistent. In both approaches, across multiple models and random seeds, `GenHlth`, `HighBP`, and `BMI` consistently appeared as the top three most important features. There were actually pretty consistent results for the top ten features across both methods with only slight variations in ordering. `Sex` did appear to be valued more by SHAP than in PFI, particularly in the XBGoost SHAP output. There was, however, one major difference in the two models regrading the feature `MentHlth`. PFI considered this feature actually hurtful to the models predictive ability, assigning it a consistent negative score across all three random seeds. SHAP, on the other hand, considered this feature just outside of the top ten most important for making predictions. More work could be done to assess how this feature impacts model performance, such as fitting new models to data that does and does not include this feature. Additionally, it may be that the instance of 1000 random, stratified samples caused over representation for this feature, making it seem more important than it generally should be.

All in all, permutation and SHAP importance largely agreed on what matters most, providing a general consistence on the most important features. However, SHAP provided richer information for understanding how and why predictions are made on a global level through its directional and importance outputs. As a result, I believe that SHAP provided the better global model output for interpretability.

Local Methods

In this section, I go over my results for LIME and Local SHAP. Because these are tests on local instances, I sampled three instances with high probability of belonging to positive class, low probability of belonging to positive class, and one example which was on the prediction threshold. This was done to gain better representation where the model is confident and unsure. The same three examples are better defined below and are the same examples used in both LIME and Local SHAP results.

LIME

Example 1

High Probability of being positive (> 0.75 chance of being positive). The true label for this example was positive.

Feature	Weight
GenHlth > 3.00	0.091
HighBP=1	0.084
BMI > 31.00	0.079
HighChol=1	0.051
CholCheck=1	0.045
HeartDiseaseorAttack=1	0.042
HvyAlcoholConsump=0	0.035
DiffWalk=1	0.028
Income <= 5.00	0.019
Stroke > 0.00	0.016
Sex=0	-0.013
MentHlth > 2.00	-0.01
Education <= 4.00	0.01
PhysHlth > 3.00	0.007

Feature	Weight
AnyHealthcare=1	0.005
NoDocbcCost=0	0.004
Fruits=0	0.003
PhysActivity=0	0.003
Smoker=1	-0.003
Veggies=1	-0.002
6.00 < Age <= 8.00	-0.001

The top five features contributing to prediction in this case were:

- General health: a value of greater than three contributed the most to making a positive prediction with a weight of 0.091**.**
- High blood pressure: the fact that this individual had high blood pressure contributed nearly as much as their bad general health to predicting the positive class with a weight of 0.084.
- BMI: having a BMI of greater than thirty-one contributed third strongest to the positive class prediction with a weight of 0.079.
- High cholesterol: the next most significant predictor was that the individual reported having high cholesterol with a weight of 0.051. This is the most significant drop in magnitude between this and the previous feature within the top five.
- Heart disease or attack: next in the top five is the individual indicating they have had a heart disease or attack. This contributed with a weight of 0.045 to the positive class prediction.

In all five case, these variables contributed to a positive outcome. Most interestingly, we see that this individual was female which contributed negatively to the positive class prediction. This directly aligns with our findings in the global model explanations.

Example 2

High probability of being negative (< 0.05 chance of being positive). The true label for this example was negative.

Feature	Weight
GenHlth <= 2.00	-0.089
HighBP=0	-0.081
Age <= 6.00	-0.057
BMI <= 24.00	-0.056
HighChol=1	0.049
CholCheck=1	0.049
HvyAlcoholConsump=0	0.036
HeartDiseaseorAttack=0	-0.027
DiffWalk=0	-0.027
7.00 < Income <= 8.00	-0.017
AnyHealthcare=1	0.008
5.00 < Education <= 6.00	-0.007
Sex=0	-0.006
MentHlth <= 0.00	0.005
Stroke <= 0.00	-0.004
PhysHlth <= 0.00	-0.004
PhysActivity=1	-0.003
NoDocbcCost=0	0.002
Smoker=0	-0.002
Veggies=1	-0.002

Feature	Weight
Fruits=0	0.002

The top five features contributing to prediction in this case were:

- General health: general health being less than or equal to two, indicating very good or excellent general health helped most in the negative prediction for this individual with a weight of -0.089.
- High blood pressure: not having high blood pressure provided a weight of -0.081.
- Age: weight of -0.057. This aligns with the global outputs that younger individuals are more likely to be negative.
- BMI: having a BMI of less than or equal to twenty-four contributed to a negative prediction with weight -0.056.
- High cholesterol: this individual did have a high cholesterol, which contributed towards a positive prediction with the model with a weight of 0.049.

While this individual is not in the positive class, the LIME output still provides actionable insights for this user: to lower their chances of diabetes, they can work on lowering their cholesterol and alcohol consumption. In this case, LIME outputs would be very beneficial as a summary for an end user like this individual.

Example 3

This instance is one where the model lacks general confidence in the prediction. This instances probability of belonging to the positive class is on the threshold used when maximizing for F1 score. The true label for this example was negative.

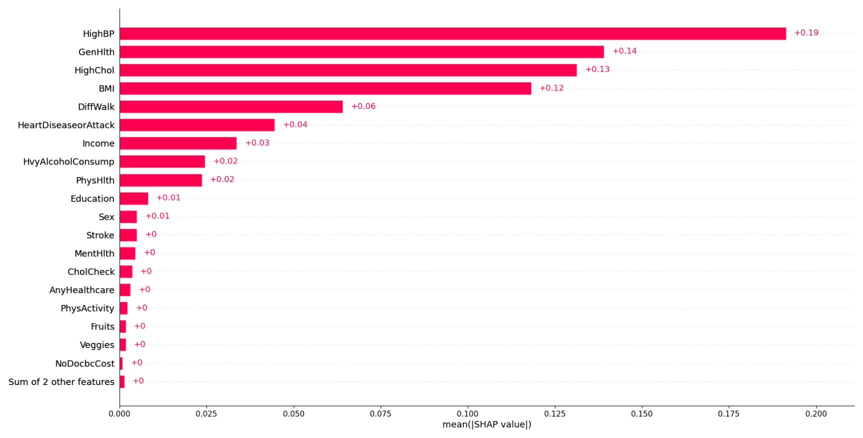
Feature	Weight
GenHlth <= 2.00	-0.089
HighBP=1	0.085
BMI > 31.00	0.078
HighChol=1	0.047
CholCheck=1	0.042
Age > 10.00	0.041
HvyAlcoholConsump=0	0.033
HeartDiseaseorAttack=0	-0.032
DiffWalk=0	-0.028
Income <= 5.00	0.016
AnyHealthcare=1	0.012
Sex=0	-0.012
MentHlth <= 0.00	0.009
Stroke <= 0.00	-0.008
Education <= 4.00	0.007
PhysHlth > 3.00	0.005
Veggies=1	-0.004
NoDocbcCost=0	-0.003
PhysActivity=1	-0.002
Fruits=1	-0.002
Smoker=0	-0.001

Interestingly, only three of the top ten features which most contributed to a prediction in this case were negative. We see in this case the model is struggling to predict an output given the target, as most of the features in this instance contribute to a positive school, even though this user is negative. In situations like this, LIME is still beneficial in warning the user that their actions are leading towards a positive prediction, even if their current outcome is negative. In such a case, this model could serve as preventative care by providing the user with direct, actionable insights on how to lower their chances for developing diabetes.

Local SHAP Feature Importance Explanations

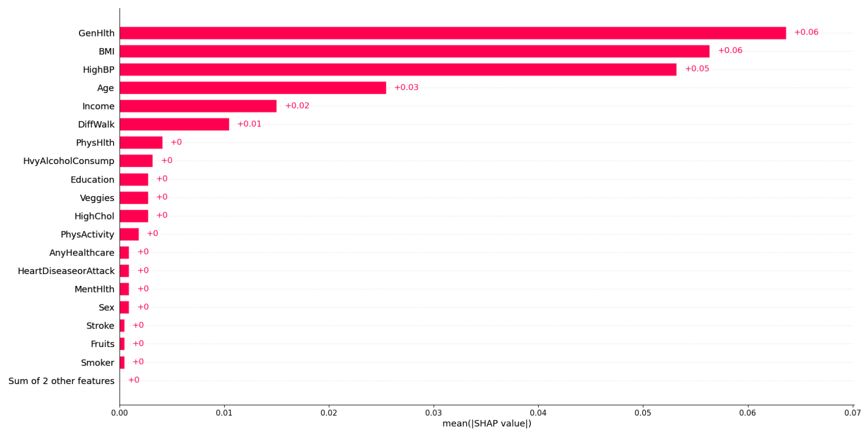
Example 1

Here is the feature importance associated with example 1. Reminder, example 1 is high probability of being positive (> 0.75 chance of being positive). The true label for this example was positive.



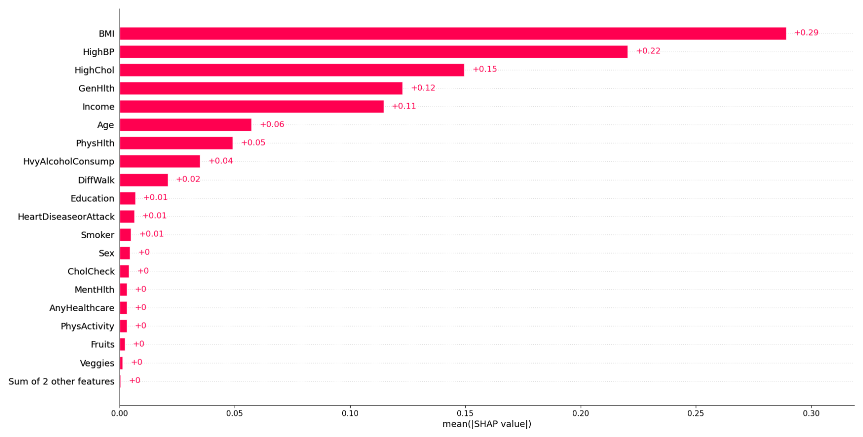
Example 2

Here is the feature importance associated with example 2. Reminder, example 2 was one with a high probability of being negative (< 0.05 chance of being positive). The true label for this example was negative.



Example 3

Here is the feature importance associated with example 3. Reminder, this instance is one where the model lacks general confidence in the prediction. This instances probability of belonging to the positive class is on the threshold used when maximizing for F1 score. The true label for this example was negative.



We still see fairly strong consistency across the top features, with an exception in example 3 where very strong emphasis was put into BMI, while general health (a clear front running in terms of most important feature across global and local tests) was undervalued.

Final Results for Local and Global Interpretability methods

This table compares the top contributing features across interpretability methods and rates the consistency of each method in identifying the most influential predictors.

Method	Top 3 Features (in order)	Consistency Score
Permutation FI	GenHlth, HighBP, BMI	High (stable across seeds)
SHAP (RF)	GenHlth, HighBP, BMI	High
SHAP (XGBoost)	GenHlth, BMI, HighBP	High
LIME	GenHlth, BMI, HighBP	Medium (some variation in each example)
Local SHAP	HighBP, BMI, GenHlth	Low (major variation across all three examples)

Global methods like Permutation Feature Importance and Global SHAP were highly consistent, while local methods like LIME and Local SHAP showed more variation. This is to be expected, since they explain only individual predictions. Overall, `GenHlth`, `HighBP`, and `BMI` were the most influential features for model prediction.

Counterfactuals

The method I used for counterfactuals is outlined in Christopher Molnar's book, specifically chapter 15 which outlines how to derive counterfactuals [2]. In this chapter, Molnar outlines two different methods for deriving counterfactuals: Wachter et al. and Dandl et al. For the project, I opted to only implement the Wachter et al. method as this is the one we covered in class and is more straightforward than the Dandl et al. counterpart.

Methodology

The objective of the counterfactual search was to find the minimal perturbation to the input features such that the model output crosses the decision threshold, flipping the predicted class. For this test, I utilized the at-home model framework, which included the at-home model features and model parameters. I also chose to only assess counterfactuals for the Random Forest, not the XGBoost. I chose to test with these conditions because counterfactuals are model agnostic and therefore would not affect the generalizability of my findings. Choosing the Random Forest over the XGBoost was simply a design decision without justification. With that, the features used were the at-home model features laid out below:

```
HOME_CATEGORICAL_FEATURES = [
    "HighBP",
    "CholCheck",
    "Smoker",
    "HeartDiseaseorAttack",
    "PhysActivity",
    "Fruits",
    "Veggies",
    "HvyAlcoholConsump",
    "AnyHealthcare",
    "NoDocbcCost",
    "DiffWalk",
    "Sex"
]
```

and the Random Forest parameters were those found during initial nest cross-validation (that process is described in the "Machine Learning Models" section of this report:

```
rf_params = {'class_weight': {0: 1, 1: 4}, 'max_depth': 10, 'n_estimators': 150}
```

To assess the counterfactuals, I randomly sampled users from a set of candidates. This set included test samples whose predicted probabilities were within a margin of the decision threshold. More specifically, the margin chosen was 0.1, meaning that if the predicted probability from the model on that sample was within the decision threshold ± the margin then the sample was added to the set of candidates. The decision threshold in this case was 0.24 meaning that if the model predicted a probability ≥ 0.24 on a sample, it was classified as positive; otherwise, it was classified as negative. This 0.24 figure was decided by choosing the threshold on a validation set's calibrated probability outputs that optimized most F1 score. Deciding on the margin value was not easy and will be discussed in more detail in a later section.

For optimizing the loss function, I used the Nelder-Mead method in the "minimize" function, which is part of SciPy's Optimization library, which was recommended by Molnar [2]. I also referenced the SciPy documentation [3]. Lastly, sometimes the optimizer would output negative values that were beyond the feature space. In a case where a negative value was output in the counterfactual, those were changed to 0. In all cases, the negative values were in the range [-0.0005, 0), so it seemed appropriate to push them to 0.

Counterfactual Results

After implementing the method, I evaluated its performance across a range of random test cases. My results using counterfactuals were very poor. For starters, many of my results were inconsistent. I expected inconsistency to some extent because counterfactuals have innumerable features that can be perturbed to minimize the loss function. For example, a counterfactual that changes features x and y is just as valid as one that changes features z and l or k and p. However, this property of counterfactuals made it difficult to assess across multiple examples and gave me pause when I considered providing them to a user, whether that be at-home or domain expert users.

I additionally found the choice of λ difficult extremely difficult. In the Wachter et al. equation, λ is a regularization parameter that penalizes deviation from the original input. Because I was unable to bound my feature values, I opted for a larger λ, as doing so would encourage the counterfactual instance to remain similar to the original feature. However, in doing so, the margin of candidates had to be lowered, as encouraging the feature to remain similar to the original on a sample that was far from the decision boundary made it difficult to push it over the decision threshold. In other words, the more meaningful I made the counterfactual—through a larger λ which ensured a counterfactual that was more similar to the sample and therefore realistic—the less candidates I could serve with the counterfactual, as I would then need to only select candidates with probability outputs close to the decision threshold. This ultimately destroyed the effectiveness of the counterfactual en masse.

I did attempt to bound my counterfactual instances to make them more realistic. To do this, I used the L-BFGS-B algorithm as my optimizer in the minimize through the SciPy Optimize library [4]. This algorithm is an extension of L-BFGS algorithm but allows the use of upper and lower bounds on variables [5]. Bounding these variables introduced the same issues as choosing a large λ described above: the bounds hinder the optimizer from modifying the instance sufficiently enough to push it across the decision boundary. As a result, the candidates for counterfactuals are drastically reduced making it available only for samples where the predicted probability is within a small window near the probability decision threshold. It is worth noting that the Dandl et al. method would have ensured realistic counterfactual instances but implementing that was beyond the scope of this project as it is much more involved. Future work could assess if the Dandl et al. method would have made my counterfactuals more user friendly.

With that, one pattern emerged in my testing that was most problematic. Many of the counterfactuals altered features not only in a way that did not make sense but also focused mainly on correlating features and not causal ones. For example, in performing over 20 counterfactuals, the majority of changes were in features `Age`, `Income`, and `Education`, while neglecting `BMI`, `PhysHlth`, and `GenHlth`, for instance. While this does provide insight into how the model assesses decisions on a local level, it is not beneficial for users who are seeking actionable insights and to understand how their lifestyle may have contributed to the model's diagnosis.

Here is an example of a counterfactual that provided little information for the subject (changes in features have been highlighted):

Feature	Original	Counterfactual
HighBP	1	1
CholCheck	1	1
BMI	55	55.04
Smoker	1	1
Stroke	0	0
Income	6	6.837
HeartDiseaseorAttack	0	0
PhysActivity	0	0
Fruits	0	0
Veggies	1	1
HvyAlcoholConsump	0	0
AnyHealthcare	1	1
NoDocbcCost	0	0
GenHlth	4	4.264
MentHlth	0	0
PhysHlth	5	5
DiffWalk	1	1
Sex	1	1
Age	9	9.011
Education	4	4.155

Sample 174904 - Current Outcome: 0 with probability 0.1211 belong to positive class - Lambda value 5

This result was one of the more interpretable examples derived. We see here that the model suggests if the user had an increase in BMI, Income, Age, Education, and general health, then the model would have classified them as at risk of being prediabetic/diabetic. We see that the model focused primarily on features correlated with the target variable, such as Age, Education, and Income, with only slight changes in causal features like BMI and general health. When the model did address causal features, it was not the primary focus as these features were only perturbed slightly. These findings highlight the importance of incorporating domain knowledge and careful interpretation of outputs before they can be presented to end users.

Challenge 3 Results: Representation

All Features

Using all the features produces the best results on F1 and model calibration. XGBoost performs the best in F1, Recall, and Balanced Accuracy with equal Brier score, making it the better expert model compared to its Random Forest counterpart when assessed on the hold-out set.

Model	F1	Recall	Precision	Balanced Accuracy	Brier Score
Random Forest	0.462	0.543	0.402	0.706	0.097
XGBoost	0.470	0.606	0.384	0.724	0.097

All Features Except `HighChol`

With the `HighChol` removed, we do see model degradation, most notably in the XGBoost model. With all features, the XGBoost model outshined the Random Forest, but upon removing `HighChol`, the model's performance was actually worse than its Random Forest counterpart, which itself also took a performance hit when this feature was removed.

More specifically, the Random Forest model decreased in quality in the two most important metrics, F1 and Brier scores. However, it's recall and balanced accuracy scores increased. Nevertheless, when comparing the two models performance on the hold-out set, the Random Forest model performed the best when the feature was removed, making this the standout model for the at-home user. Below is the model output after removing `HighChol`.

Model	F1	Recall	Precision	Balanced Accuracy	Brier Score
Random Forest	0.461	0.602	0.374	0.719	0.098
XGBoost	0.460	0.597	0.375	0.717	0.098

This output will be treated as the baseline throughout the rest of feature removal evaluations.

`HighChol` & Calculation Features Removed

Reminder, 'Calculation; features are the following:

- `Income`: this feature requires the user to input their age as described in the dataset's codebook. This feature is on a scale from 1 to 8 with 1 being less than $10,000, 5 being less than $35,000, and 8 being $75,000 or more, for example.
- `Education`: this feature requires the user to input their education level as described in the dataset's codebook. This feature is on a scale of 1-6 where 1 is never attended school or only kindergarten, etc.
- `Age`: this feature requires the user to input their age as described in the dataset's codebook. This features is on a scale of 1-13, with 1 being 18-24 years of age, 9 being 60-64 years of age, and 13 being 80 or older, for example.

Removing these features hurt model F1 score and calibration when compared with the baseline metrics above.

Model	F1	Recall	Precision	Balanced Accuracy	Brier Score
Random Forest	0.445	0.557	0.370	0.702	0.100
XGBoost	0.451	0.596	0.363	0.713	0.100

In an effort to conform more to the baseline, I re-ran the model but included the `Age` feature. Here are the results now with `Income`, `Education` & `HighChol` features removed:

Model	F1	Recall	Precision	Balanced Accuracy	Brier Score
Random Forest	0.458	0.584	0.376	0.714	0.099
XGBoost	0.462	0.637	0.363	0.728	0.098

These results are competitive with—and in the case of the XGBoost model, better than—the baseline results. Therefore, I conclude that while removing `Age` from the feature set does lower the barrier for use, this feature increases the F1 score by over 10% in both models and improves calibration, making it a tolerable barrier for the at-home user.

`HighChol` & Health Survey Features Removed

Reminder, 'Health Survey' features are the following:

- `MentHlth`: this features asks the user to provide the number of days of poor mental health over the past 30 days.
- `PhysHlth`: this feature asks the user to provide the number of days that they experienced physical illness or injury in the past 30 days.
- `GenHlth`: this asks the user, quite vaguely, to assess their general health on a scale of 1-5 with 1 being excellent, 2 very good, 3 being good, 4 fair, and 5 poor.

Removing these features actually hurt my model evaluations with respect to F1 and calibration more so than removing the calculation features described above. The results from removing `HighChol`, `MentHlth`, `PhysHlth`, & `GenHlth` are:

Model	F1	Recall	Precision	Balanced Accuracy	Brier Score
Random Forest	0.442	0.598	0.350	0.709	0.101
XGBoost	0.444	0.619	0.346	0.715	0.101

To further assess this behavior, I re-ran the tests but kept the `GenHlth` feature and obtain the following results:

Model	F1	Recall	Precision	Balanced Accuracy	Brier Score
Random Forest	0.460	0.603	0.372	0.719	0.098
XGBoost	0.465	0.610	0.376	0.723	0.098

By adding back in this feature, I was able to obtain evaluations that we just as good or in some cases better than simply removing `HighChol`. In other words, by removing `HighChol`, `MentHlth`, & `PhysHlth` features but keeping `GenHlth`, my model performed just as well as the baseline (all features except for `HighChol`) on the hold-out set.

I conclude that by removing the `MentHlth` & `PhysHlth` features, I am able to maintain good performance while lowering the barrier of use for at-home users.

Combining Results

During model evaluation, I found that removing the `HighChol`, `Income`, `Education`, `MentHlth`, & `PhysHlth` provided most competitive outputs compared to the baseline of just removing `HighChol`. However, these were assessed separately with `HighChol`, `Income`, & `Education` being removed in one test and `HighChol`, `MentHlth`, & `PhysHlth` being removed in the other. In this section, I have removed all five features and obtained the following metrics on the hold-out set. I have also included the baseline metrics in parenthesis for easy-to-see comparison. Recall that the baseline was each model with all the features except `HighChol`:

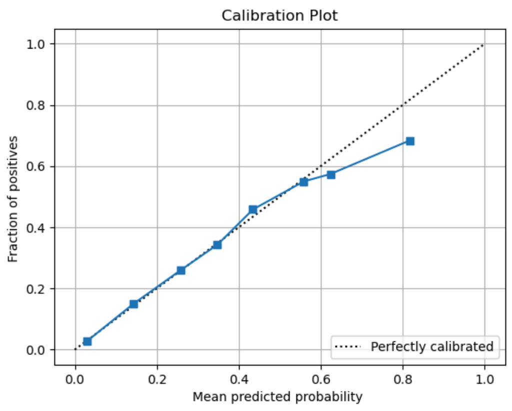
Model	F1	Recall	Precision	Balanced Accuracy	Brier Score
Random Forest	0.460 (0.461)	0.597 (0.602)	0.374 (0.374)	0.718 (0.719)	0.099 (0.098)
XGBoost	0.465 (0.460)	0.629 (0.597)	0.368 (0.375)	0.727 (0.717)	0.098 (0.098)

Baseline metrics (Model with all features except `HighChol`) in parenthesis

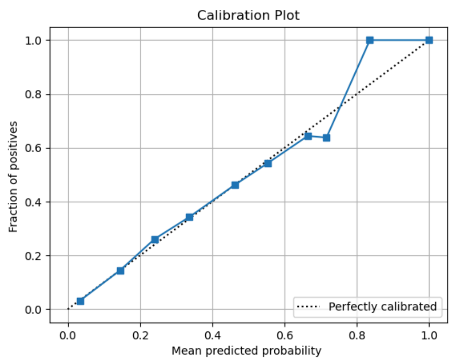
The F1 Scores in both cases are within 5% of one another and have nearly identical Brier scores. To see model calibration, here is a calibration plot for the baseline and this model:

Baseline Calibration Plot (Only `HighChol` Feature Removed)

Random Forest

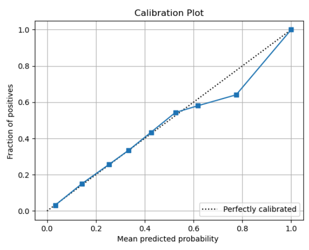


XGBoost

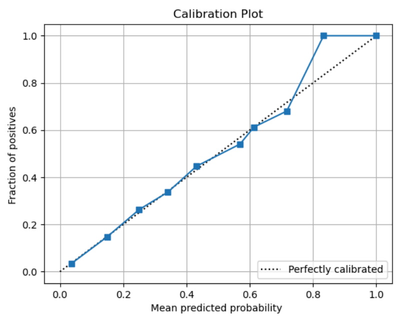


Updated Model Calibration Plot (`HighChol`, `Income`, `Education`, `MentHlth`, & `PhysHlth` Features Removed)

Random Forest



XGBoost



Model calibration with the additional four features removed is nearly identical to that of the baseline. The results of these two models, with four extra features removed, is competitive with the standard at-home baseline models, making it the best configuration for the at-home user by effectively lowering the threshold for use while not making major sacrifices in model performance.

7. Reflection

What surprised me most was my results for challenge 3. I did not think I would be able to remove these features, while still obtaining such competitive results with a model using all features except `HighChol`. In other words, I was able to remove `Income`, `Education`, `MentHlth`, & `PhysHlth` and still obtain great results while lowering the threshold for users. After looking at my global feature importance results, it makes some sense that I was able to remove these, as they were not the most important features overall. However, going in, I thought I would see major drops in performance by removing features.

I showed my model's performance results to Frankie Hodges. He was mainly surprised that my recall/F1 scores were so low, even with such a massive dataset. He expressed that he found it difficult to be satisfied with his recall and F2 results because he expected them to be higher. I am personally not too surprised by this. It seems that any time you are dealing with a complex system like human health, it's difficult to obtain an exhaustive set of features that allow you to explain most of the target variance, even with large amounts of data and an expressive tree model.

He did provide constructive feedback, particularly regarding my use of F1 score, stating that using F2 would have allowed me to prioritize recall. This gets to the heart of a more difficult challenge I faced during this project: how do you threshold probability outputs such that you properly balance recall and precision. There didn't seem to be a clear answer to me. With more time, I would have preferred to consult with a domain expert and get their insight.

A few things changed in my final project from my draft to my final results. Particularly, I found that I hadn't considered thoroughly some of the metrics I originally proposed. I also think that I didn't fully understand some of the interpretability methods I originally considered using. For example, I had originally said I was going to use Partial Dependency Plots for interpretability but when I sat down to read prior to implementation, I found that this method would not yield desired results. I also initially set too broad of a scope regarding things I wanted to address and the various ways to do so. In the future, I plan to dial back a bit, establish a few methods, and do them each really well.

With more time, I could dive further into each challenge. I don't think I adequately was able to compare the Random Forest and XGBoost model. I found that they performed similar in most cases with only a few outlier situations. I would like to have explored those more. Additionally, so much more can be done with interpretability. I didn't realize how large of an undertaking this would be and how many different directions there were to explore. The amount of comparison you can make between these methods felt endless. Finally, and most importantly, I would have liked to understand the possible impact of a model like this. I would have loved to get feedback from non-technical users, and explored how they felt about each interpretability method. I also would have liked to speak with a domain expert about proper thresholding and get their take on the model's output.

All in all, this project ended up taking me a lot more time than I thought. I found the interpretability methods to be surprisingly difficult to assess and computationally expensive, but also exceedingly useful in understanding how the model made it's decisions. I do feel that some components could benefit from deeper exploration given more time, but overall, I learned a great deal from this project.

R. Response to Draft Report Feedback

Feedback Point: Show what the possible values are for each categorical feature.

Response: I added a column showcasing the possible values for each categorical feature

Feedback Point: Add a table header to your data examples giving the index of each example shown.

Response: I added a table header to my three data examples giving the index of each example.

Feedback Point: While an RF in theory can handle categorical variables, the one in sklearn does not. Clarify whether you are using a different implementation (that can), or omit this justification

Response: Justification omitted as I was unaware that sklearn's Random Forest model does not handle categorical features natively.

Feedback Point: Add citations for LIME, the method you used for counterfactuals, permutation feature importance, etc.

Response: Added the citations for LIME, counterfactuals, and permutation feature importance, as well as some other method mentioned.

Feedback Point: Provide more details about your hyperparameter search. Hyperparameter tuning needs to be done on a subset of the data that is disjoint from your eventual test set. This usually means splitting into train/validation/test and doing your hyperparameter selection by maximizing performance on the validation set, or CV inside the training set. In particular, it is not valid to optimize hyperparameters and then re-do CV on the same data set (this is a kind of data leakage).

Response: I made sure not to leak data by using the same train, test, validation split sizes and random seed across all instances. This way, the same test dataset stripped before training. I included a paragraph stating this in my report.

Feedback Point: Based on the description here, it's not clear what is different between the at-home and expert models when doing hyperparameter search. Assuming they are using the same data, explain why different parameters were selected. (So far the only explanation of the differences in models is that the at-home model will only give a classification while the expert model also gives a calibrated probability.

Response: I added a paragraph in my report explaining the key difference between these two models. In summary, these models differ in features. The `HighChol` feature was found to be important for model prediction, but is difficult to assess for an at-home user. As a result, I removed this feature for the at-home user. This is why separate models and parameters were selected.

Feedback Point: Explain how you computed "balanced accuracy".

Response: This was done using Scikit-Learn’s built in metric. I added that information to the report with a citation to their docs.

Feedback Point: Explain what you mean by "calibrated probability outputs" and what kind of metric this is. (It's probably not a metric but rather a strategy you will try later. Maybe ECE is what you meant here.)

Response: I added a section on model calibration to make my methodology more clear, including the method used.

Feedback Point: The confusion matrix is an inspection tool but not a metric. A metric is a value you can measure

Response: Fair point. I removed the reference to confusion matrices as a metric.

Feedback Point: As noted above, this approach will cause data leakage: "After selecting the best performing configuration of parameters, I performed standard a standard train/validation/test split of 60%/20%/20% respectively." You should do the hyperparameter search only on the train (or train/val) data.

Response: I should have been more clear that the split here is the same one done during hyperparemeter tuning in which I only tuned on the training set using nested-cross k-fold validation. In other words, the validation and testing set is the same one across both splits and remained untouched during hyperparmeter tuning. I have updated the language in my report to clarify this confusion.

Feedback Point: Use majority class prediction as a baseline, especially with imbalanced data. What you described is useful but it is not a baseline.

Response: I added a majority class prediction as a baseline output for challenge 1.

Feedback Point: Class imbalance: ensure that any resampling or SMOTE is done only on the training set (test set should stay fixed for all strategies).

Response: The test set did remain constant and untouched throughout the project. SMOTE was not performed on the test set. I have made that more clear in my report.

Feedback Point: Add a citation for LIME, SHAP, PFI, etc. Be more specific about what you actually do, not "such as". More details about these methods as most have parameters you need to specify or other details to make them work on your data (Instructions: "Explain in enough detail that someone else could use this strategy")

Response: I made sure to add the citations. I added more information about what libraries I used and explained the methodology more in the results section.

Feedback Point: Cost-sensitive learning: Explain what penalty you used.

Response: The exact class weights have been added so that it is more clear what was used.

Feedback Point: Feature selection: More details are needed for forward/backward selection (how many will you select?). Also explain why you would use different metrics for the at-home and expert models

Response: This section evolved a bit as I realized that my feature selection had to be more intelligent than simply maximizing metrics. My goal for the representation challenge was to lower the threshold for the at-home user to obtain a prediction. Therefore, instead of doing features selection that maximized a metric, I chose to identify 7 key features that were more difficult to input than others. I then went through an iterative process of removing these in various combinations to see if I could obtain model evaluations that were competitive with the standard at-home model, while having less 'difficult' features. I have updated the report to reflect this change in strategy.

Feedback Point: Include the majority-class baseline result in each table.

Response: I ended up providing the majority-class baseline results in a separate table. This is now added to the report.

Feedback Point: I thought you would have 4 results for each challenge since you are aiming for RF and XGB for at-home and expert use. Explain which one we are seeing, and add the other one (if that makes sense).

Response: Because the expert model and the at-home model only differ by the removal of one feature, I opted to look at class imbalance with only the expert model, as that is the one that contains all the features. When I ran the tests with the at-home model, the underlying behavior was the same with a drop in evaluation across the board. However, I did not feel like this was important to include as the consistent drop in metrics seemed to be more of a representation issue instead of class imbalance.

Feedback Point: Many of your metrics are not included (recall, precision, accuracy, balanced accuracy) - add them in. You may find this gives you more to discuss.

Response: I have updated the metrics I chose to use to F1, recall, precision, and balanced-accuracy. These are now included in the report.

Feedback Point: Check whether the differences between cost-sensitive learning and "do nothing" are significant and include this in your analysis of the results.

Response: The differences on the hold-out set was not significant—they basically performed the same. However, I believe that the cost-sensitive learning class weights will generalize better on new data. I provided a paragraph justifying this claim in my report**.**

Feedback Point: Think carefully about what AUC is telling you and whether it is useful for this problem. For example, for AUC, "undersample" was tied with cost-sensitive learning. This should be discussed.

Response: After some consideration, I decided to remove AUC as a metric in this report. I found it difficult to interpret and that it did not provide any additional insight beyond my current metrics.

Feedback Point: Correct the citation - it mentions Elon and Averbuch-Elor (2022) but the citation is to van den Goorbergh et al. (2022).

Response: Yes! Thank you! I just started using Zotero and my lack of experience with it is already creating problems. This is now fixed.

References

[1] Y. Elor and H. Averbuch-Elor, "To SMOTE, or not to SMOTE?," *arXiv.org*, May 11, 2022. https://arxiv.org/abs/2201.08528

[2] Molnar, C., "Interpretable Machine Learning: A Guide for Making Black Box Models Explainable (3rd ed.)," 2025. christophm.github.io/interpretable-ml-book/

[3] "scipy.optimize.minimize — SciPy v1.9.3 Manual," *docs.scipy.org*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html#scipy.optimize.minimize

[4] "minimize(method='L-BFGS-B') — SciPy v1.14.1 Manual," *Scipy.org*, 2024. https://docs.scipy.org/doc/scipy/reference/optimize.minimize-lbfgsb.html

[5] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B," *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, Dec. 1997, doi: https://doi.org/10.1145/279232.279236.

[6] "sklearn.inspection.permutation_importance," *scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.inspection.permutation_importance.html

[7] S. Lundberg, "An introduction to explainable AI with Shapley values — SHAP latest documentation,"*Readthedocs.io*, 2018. https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%

[8] SHAP, "Welcome to the SHAP Documentation — SHAP latest documentation, *shap.readthedocs.io*, 2025. https://shap.readthedocs.io/en/latest/

[9] "shap.Explainer — SHAP latest documentation,"*shap.readthedocs.io*. https://shap.readthedocs.io/en/latest/generated/shap.Explainer.html

[10] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 1135–1144. https://doi.org/10.1145/2939672.2939778

[11] "sklearn.metrics.balanced_accuracy_score — scikit-learn 0.21.3 documentation," *Scikit-learn.org*, 2010. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html

[12] "1.16. Probability calibration," *scikit-learn*. https://scikit- learn.org/stable/modules/calibration.html

[13] "SMOTENC — Version 0.11.0," *imbalanced-learn.org*. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTENC.html