CIS520 Koala Kubs
Final Project - Binary Classification Models for Type 2 Diabetes

Seung-Hyun Ko, Kshitiz Parihar, Yi Qi
UPenn
sbko@seas.upenn.edu, kparihar@seas.upenn.edu, yqi01@seas.upenn.edu

Abstract

As much as 45% of the U.S. population is affected by diabetes, a chronic disease in which the body's inability to break down sugars causes an increased concentration of bloodstream sugar. Over time, this elevated blood sugar can lead to fatal conditions such as heart or kidney disease, motivating the use of models to identify risk factors for diabetes. Using responses to the CDC's Behavioral Risk Factor Surveillance System from 2011-2015, here we optimize and compare several binary classification models' performance in accurately predicting diabetic status using a curated set of questions from the survey. During optimization, we explore several methods for addressing class imbalance, and from our results we find that a light gradient boosting machine had optimal performance classifying diabetic and non-diabetic samples. We then use this optimal model to identify trends in feature importance across the years and find that the most important predictors of diabetic status remain constant.

1. Introduction

Motivation. Diabetes is a chronic disease in which the body is unable to properly convert sugars into usable energy for cells, resulting in an increased concentration of sugar in the bloodstream. Over time, this elevated bloodstream sugar concentration can lead to serious health concerns, such as heart or kidney disease. According to the CDC, as of 2020 34.2 million Americans (10.5% of the population) have diabetes and another 88 million adults (34.5% of adult population) have pre-diabetes [5]. Given the serious health risks of diabetes, it is important to identify risk factors associated with diabetes so that they can be addressed as early as possible to prevent disease onset/progression. Machine learning can be a powerful tool to identify these risk factors, especially given the biological, social, and economical diversity of the American population.

Related Work. In 2019, Xie et al. [11] used the 2014 Behavioral Risk Factor Surveillance System to investigate potential risk factors for type 2 diabetes. The study compared several machine learning models in terms of their prediction of type 2 diabetes. Models included support vector machine, decision tree, logistic regression, random forest, neural network, and Gaussian Naive Bayes classifiers. This study concluded that the neural net model, though having the highest accuracy, AUC, and specificity scores, had the lowest sensitivity. Meanwhile, the decision tree model had the highest sensitivity score. Furthermore, they used adjusted odds ratio to analyze different risk factors of the disease and concluded that respondents who had sleeping time more than 9h/day and the last checkup within a year had higher risk of developing type 2 diabetes.

2. Dataset

The CDC conducts an annual Behavioral Risk Factor Surveillance Survey (BRFSS) to collect state-specific data about US residents' general health. In each state, adults are randomly selected to be surveyed over phone, resulting in usually 400,000+ responses to several hundred questions each year. On Kaggle, the CDC provides annual data for 2011-2015 [4], and using the 2015 data Alex Teboul made a cleaned up, diabetes-specific dataset [9]. This cleaned diabetes dataset has dimensions of 253,680 x 21 (responses x questions), with 213,703 non-diabetic, 4,631 pre-diabetic, and 35,346 diabetic respondents; we chose to merge the pre-diabetic and diabetic classes for ease of calculating evaluation metrics and the help alleviate the class imbalance somewhat. Lastly, the CDC provides annotations mapping encoded to real responses (eg. for question about state of residence, 1: Alabama, 2: Alaska, etc.) to facilitate analysis of the data [3].

2.1. Data Pre-processing

2.1.1 Feature Selection

After identifying the union of the features selected by Alex Teboul from the 2015 dataset and by Xie et al. from the 2014 dataset, 24 features were shared in the dataset across all years. We dropped the feature *'TOTINDA'* because its values were defined based on the values of feature *'EXERANY2'*. *'DIABETE3'* was the response variable in our study, leaving us with 22 features for analysis.

2.1.2 Feature Engineering

In the original CDC BRFSS data, *'DIABETE3'* (our target variable) had 3 classes: non-diabetes, pre-diabetes, and diabetes. We decided to change this variable to a binary one by merging two categories for the convenience of further analysis. A classification model predicting a pre-diabetic person as diabetic is more consistent and safe from a health perspective compared to classifying them as non-diabetic. Hence, we merged pre-diabetes with the diabetes class.

2.1.3 Data Cleaning

We followed the data cleaning protocol made by Alex Teboul, wherein all samples with a missing value (NaN) in any of the selected features are dropped and samples with survey answers of the type *'don't know'*, *'not sure'*, or *'refused to answer'* were dropped. This protocol was applied to all the years.

2.2. Data Summary and Visualizations

The number of responses left for each of the years after data cleaning is summarized in Table 1. There are roughly 170,000 to 310,000 respondents left in each year post data cleaning.

Year	2011	2012	2013	2014	2015
Num Resp.	312575	281031	254791	213701	169386

Table 1. Number of respondents left after data cleaning

Given that the domain of the problem in the real world is highly imbalanced (i.e. *diabetic* is much rarer than *non-diabetic*), the data for each of the years is also highly class imbalanced with roughly 1 out of 6 respondents being in the *diabetic* class. Figure 1 summarizes this class imbalance for each year.

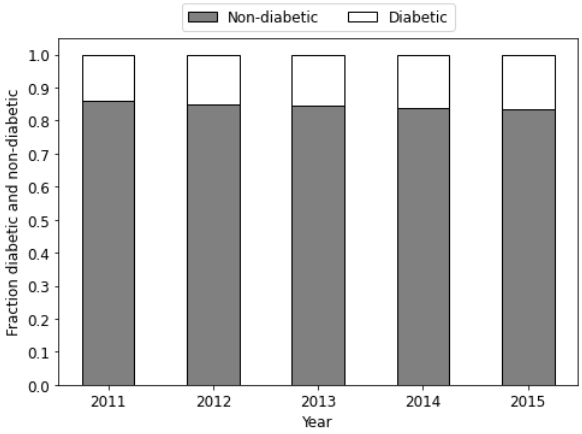


Figure 1. Class Imbalance

The final set of 22 selected features fall under four types, namely ordinal (clear ordering of the categories), nominal (no intrinsic order of categories), binary (subset of nominal, yes or no) and real/distinct valued as shown in Table 2. Most of the features are binary (for example, *SMOKE100: Have you smoked at least 100 cigarettes in your entire life?*). The prominent type is ordinal (for example, *INCOME2: Is your annual household income from all sources?* has categories 1-8 with 1 being <10k, 2-7 for intermediate income ranges and 8 being >75k).

Ordinal	CHECKUP1; EDUCA; GENHLTH; INCOME2; MSCODE; _AGE5YR; _SMOKER3
Nominal	MARITAL, RENTHOM1
Binary (0: 'No'; 1: 'Yes')	ADDEPEV2; CHCKIDNY; CVDCRHD4; CVDSTRK3; EXERANY2; HLTHPLN1; MEDCOST; SEX; SMOKE100; USEEQUIP
Real or distinct valued	_BMI5 (real valued); PHYSHLTH , MENTHLTH (value: number of days 0-30)

Table 2. Types of features

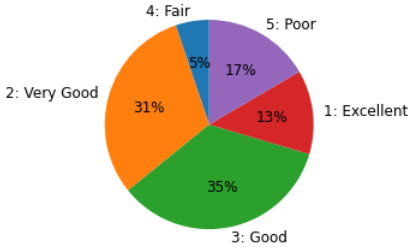


Figure 2. Respondent distribution for GENHLTH (ordinal feature) in 2015 dataset

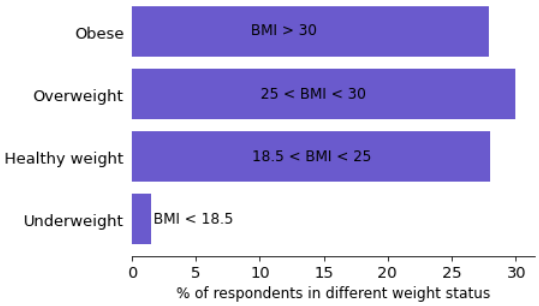


Figure 3. Respondent distribution for _BMI5 (Real valued) in 2015 dataset

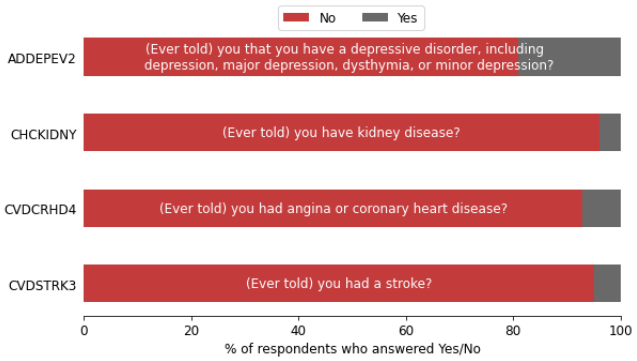


Figure 4. Respondent distribution for some binary features in 2015 dataset

For the cleaned 2015 dataset, figure 2 shows the percentage of respondents in 2015 dataset in different categories of the ordinal feature GENHLTH (Would you say that in general your health is?). Figure 3 shows the distribution of respondents in 2015 respondents in different weight status. In figure 4, we can see what fraction of respondents answered yes (or no) in the 2015 dataset for some of the health related questions.

3. Problem Formulation

After data pre-processing, we have 22 features and 1 target variable specifying the diabetic status. We used only the cleaned 2015 dataset for training and testing of models, as combining all the 5-year dataset would be too computationally expensive for training. Additionally, the cleaned 2015 dataset was assumed to be representative of the combined five year data since the class imbalance in all the years is similar (Fig.1).

Several binary classification models are explored and evaluated to select the best model for further use. Following this, we check the predictive capabilities of the best model in different years (2011-2014) and analyze the relative importance of features across the years (2011-2015) using SHAPley analysis to compare how the set of most important features changes with time.

The cleaned 2015 dataset has 169,386 samples and 22 features. The binary target variable is defined as *diabetic* $y = 1$ and *non-diabetic* $y = 0$. We split the cleaned 2015 dataset into train and test sets (80-20), ensuring that the class imbalance in the two sets is same as that in the original cleaned 2015 dataset. Table 3 summarizes class sizes in the train and test sets.

	Class 1 (diabetic)	Class 0 (non-diabetic)	Total
Train Set	22712	112796	135508
Test Set	5678	28200	33878

Table 3. Class sizes in train and test sets

4. Methods

Our **baseline model** is logistic regression with symmetric cost function. This model will be used to benchmark our other, more complex models to see if we can achieve notable improvement in performance with these models.

4.1. Logistic Regression

Logistic regression is a type of generalized linear model in which the output is some linear combination of the model inputs and parameters. It utilizes the logistic function (Eqn.2) to estimate a given observation's probability, h_θ, of being in a class (probability $1 - h_\theta$ of being in other class), and based on a threshold value these probabilities are converted into discrete class labels.

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{1}$$

$$\mathcal{L}(h_\theta(x), y) = \frac{1}{n} \sum_{i=1}^{n} \Big(- w_1 \cdot y_i \log(h_\theta(x)) - w_0 \cdot (1 - y_i) \log(1 - h_\theta(x)) \Big) \tag{2}$$

$class\_weights$ = 'diabetic' : w_1, 'non-diabetic' : w_0
Package: *sklearn.linear_model.LogisticRegression*

To introduce asymmetry, we tuned the *class_weight* parameter to alter the weights associated with the two classes, w_0 and w_1. In baseline/symmetric logistic regression, $w_0 = w_1 = 1$; by altering this we are able to penalize misclassifications in one class more harshly than the other, allowing us to address the class imbalance and also place emphasis on avoiding false negative diagnoses of diabetes. We also tune the regularization parameter *C*.

4.2. Random Forest

Random forest is ensemble method where numerous shallow decision trees (base learners) are used and the final classification is made based on majority voting. It involves bagging, i.e. using subsets of the training data, obtained either with or with out replacement, for training trees.
Package: *sklearn.ensemble.RandomForestClassifier*

We used bagging with replacement (bootstrap) for training. To introduce asymmetry among classes, we used *class_weight* parameter to assign weights associated with the two classes.

4.3. Gradient Boosting

Gradient Boosting is an ensemble method based on the idea of training weak learners (decision trees) sequentially (in a stage-wise manner) where each learner tries to correct the errors made by its predecessor. The ensemble output is a weighted average of the weak models. We compare the performance of three optimized gradient boosting methodologies, namely XGBoost, LightGBM and CatBoost.

4.3.1 XGBoost (eXtreme Gradient Boosting)

A powerful distributed gradient boosting framework which has been optimized to offer fast computational speeds and better performance compared to several other implementations of gradient boosting [2].
Package: *xgboost.XGBClassifier*

We used the learning objective as *'binary:logistic'* which has the same log-loss (cost) function as logistic regression for binary classification (eqn. 2). For this method, asymmetry in the cost function is introduced through the hyperparameter *scale_pos_weight* which can be seen as the weight for positive class (i.e. diabetic in our case).

4.3.2 LightGBM (Light Gradient Boosting Machine)

Another optimized distributed gradient boosting framework which differs from XGBoost primarily in the way it constructs trees. In XGBoost, the trees grow row-by-row (level-wise), while LigthGBM grows trees leaf-wise [12]. It has also been argued that LightGBM is faster than XGBoost [12].
Package: *lightgbm.LGBMClassifier*

We used the learning objective as *'binary'* which has the same log-loss (cost) function as logistic regression for binary classification (eqn. 2). Similar to XGBoost, the asymmetry in the cost function for this method can be incorporated using the hyperparameter *scale_pos_weight*.

4.3.3 CatBoost (Categorical Boosting)

Like the previous two gradient boosting frameworks, it has also been optimized for speed and performance. Additionally, CatBoost has in-built capabilities for working specifically with categorical features. Since 19 out of the 22 features in our dataset are categorical, we were interested in seeing how this model performs compared to other two boosting methods.
Package: *catboost.CatBoostClassifier*

We used the learning objective as *'Logloss'* which is the same cost function as used in logistic regression for binary classification (eqn. 2). Similar to XGBoost, the asymmetry in the cost function can be incorporated using the hyperparameter *scale_pos_weight*.

4.4. Neural Nets

We constructed a fully connected neural network with 10 hidden layers with varying sizes for binary classification, and the model structure is shown below. Each layer was activated by ReLU function.
Package: *torch.nn.Module*

The criterion used in this model is the *CrossEntropyLoss* implemented by PyTorch. In binary classification, this loss function is the same as that used in logistic regression (eqn. 2), and it also introduces a weight parameter to tackle class imbalance by penalizing the more abundant class.

5. Experiments and Results

5.1. Evaluation Metric

Since the dataset is highly imbalanced, we cannot use accuracy as an evaluation criteria for a classification model. Our classification goal is to correctly predict diabetic cases (i.e. high accuracy for the minority class or high precision) with the constraint that cost associated with false negative (i.e. missing a diabetic case) is very high. Therefore, we need a high recall value. Since F1-score captures both precision and recall, we selected F1-score as our evaluation metric for tuning hyperparameters and comparing different classification methods.

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}$$

Here *TP*, *FN*, *TN* and *FP* are True Positive, False Negative, True Negative and False Positive, respectively. For this study, positive means to *diabetic* $(Class = 1)$ while negative refers to *non-diabetic* $(Class = 0)$.

5.2. Class Imbalance

- We focused on algorithm-level methods to account for the class imbalance. More specifically, we used asymmetric loss (cost) function by assigning a higher weight to the *diabetic* class. This makes a false negative prediction more costly and thereby, improve the performance [6].

- Another way to handle class imbalance is by calculating an optimal threshold instead of using the default value of 0.5 for binary classification [1]. We calculated the optimal threshold value for a given trained model as the point on the precision-recall curve that has the highest F1-score.

- In data-level methods, the dataset can be balanced by either by undersampling (remove examples from majority class) or by oversampling (generate new samples for minority class) [6]. However, one has to be cautious about resampling the dataset if the imbalance between the classes is due to the fact that in reality one class is inherently rarer than the other. In this case, it can be argued that critical information of intrinsic class imbalance in the problem domain is lost due to resampling [8]. With this in mind, we tried undersampling for training some of the models considered in this study. However, this did not improve performance more than the combination of asymmetric cost and optimal threshold (Table 9).

5.3. Hyperparameter optimization using HyperOpt

For logistic regression and neural nets, *GridSearchCV* was used for hyperparameter tuning. In the case of ensemble methods (random forest and gradient boosting), we used a bayesian

optimization based python library called *hyperopt* for hyperparamter tuning [10]. It converges to the optimal hyperparameter set in a fast and efficient way by using the information it has gained from the visiting different points (bayesian approach) in hyperparamter space to decide which points to explore next. It allows for a more automated and much wider search over the parameter space as opposed to what could be achieved by manually tuning hyperparameters to explore in grid search or random search. These characteristics of hyperOpt based hyperparameter tuning especially helps in boosting classifiers, which have a large number of hyperparamters, to obtain more fine tuned values than possible via manually testing certain values in *GridSearchCV* or via random sampling in *RandomizedSearchCV*.

5.4. Classification Models

We compared logistic regression, Random Forest (RF), gradient boosting (XGBoost, LightGBM and CatBoost), and neural network (NNs) binary classification models for predicting diabetic status. For all the classification models, the analysis pipeline followed is

1. Try the model with default parameters
2. Add class weights for asymmetric loss function
3. Find optimal hyperparameters based on 5-fold cross validation using GridSearchCV or HyperOpt
4. Calculate optimal threshold value using precision-recall curve for the model trained on optimized hyperparameters

5.4.1 Logistic Regression

Table 4 lists the mean F1-scores for 5-fold cross validation evaluated for different logistic regression models tried.

	F1-score
Baseline	0.247
Tuned Asymmetric	0.468
Optimized Asymmetric	0.468

Table 4. 5-fold cross validation mean F1-scores for Logistic Regression

Our baseline was a symmetric (i.e. equal class weights) logistic regression model, and no hyperparameter tuning or threshold optimizing was performed. The model's mean F1-score for 5-fold cross-validation on the training set was 0.247 and 0.248 on the test set (Fig.5, Table 9), resulting in a high number of false negative predictions of diabetes. We consequently try to improve upon the baseline model by introducing asymmetric class weights and finding an optimal class probability threshold.

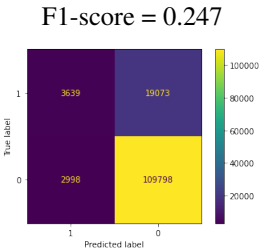


Figure 5. Confusion matrix for baseline (symmetric) logistic regression model performance on the train set.

We used *GridSearchCV* to search over our hyperparameter grid and tuned our model using the F1-score to identify optimal hyperparameters (Fig.6).

```
param_grid =
{'C': [1e-4, 1e-2, 1, 1e2],
 'class_weight': [{0:1, 1:2}, {0:1, 1:3},
                  {0:1, 1:4}, {0:1, 1:5},
                  "balanced"]}
```

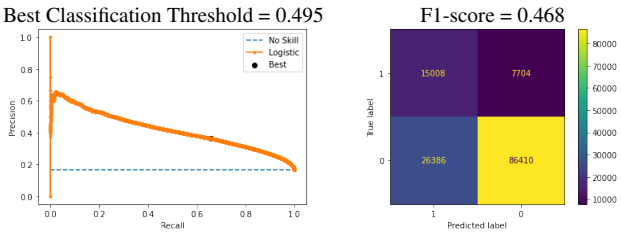
(a) Regularization and class weight hyperparameter grid.

```
{'C': 0.01,
'class_weight': {0: 1, 1: 4}}
```

(b) Optimal hyperparameters.

Figure 6. Hyperparameter tuning for logistic regression model.

With this tuned model, we saw an increase in 5-fold cross-validation performance as shown by the F1-score of 0.468 (Table 4). We then used a range of classification thresholds with this model, generating a precision-recall curve to identify an optimal threshold based on F1-score (Fig.7a). Finally, we used this threshold to evaluate our tuned asymmetric model performance on the train set. (Fig.7b).



(a) Precision-recall curve

(b) Confusion matrix

Figure 7. Logistic regression model trained on optimal hyperparamter set: (a) Precision-Recall curve with the the optimal threshold value and (b) Confusion Matrix (using optimal threshold) on the training data

5.4.2 Random Forest

Table 5 lists the mean F1-scores for 5-fold cross validation evaluated for different random forest models tried.

	F1-score
Default (max_depth = 3)	0.007
class_weight = 'balanced' (max_depth = 3)	0.444
Optimal hyperparamters	0.459

Table 5. 5-fold cross validation mean F1-scores for Random Forest

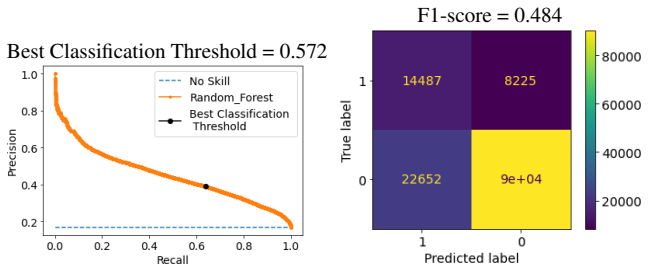
Here, in 'balanced' mode for *class_weight* parameter, the weights associated with each class are automatically adjusted to (Total number of samples) / ((number of classes) * np.bincount(y[class])). i.e. weights are taken as inversely proportional to class frequency in the input data. The hyperparameters used in the tuning process were

```
rf_param_space =
{'max_depth': hp.choice('max_depth', range(2,9)),
 'max_features': hp.choice('max_features', ['sqrt','log2']),
 'min_samples_split': hp.choice('min_samples_split',range(2,8)),
 'n_estimators': hp.choice('n_estimators', range(100,500)),
 'class_weight': hp.choice('class_weight',
                           ['balanced','balanced_subsample'])}
```

The 'balanced_subsample' mode is same as 'balanced' except that the weights are now adjusted for each tree grown individually based on the training subset (bagging + bootstrap) used for that tree. The optimal hyperparamter values obtained for random forest are

```
{'class_weight': 'balanced',
 'max_depth': 8, 'max_features': 'log2',
 'min_samples_split': 6, 'n_estimators': 207}
```

The best classification threshold value and the performance on the whole training set for the random forest model trained on optimal hyperparamter set (with predictions made based on the optimal threshold) is summarized in figure 8.



(a) Precision-Recall curve (b) Confusion matrix

Figure 8. Random Forest model trained on optimal hyperparamter set: (a) Precision-Recall curve with the the optimal threshold value and (b) Confusion Matrix (using optimal threshold) on the training data

5.4.3 XGBoost

Table 6 lists the mean F1-scores for 5-fold cross validation evaluated for different XGBoost models tried.

	F1-score
Default	0.245
scale_pos_weight = 6	0.449
Optimal hyperparamters	0.476

Table 6. 5-fold cross validation mean F1-scores for XGBoost

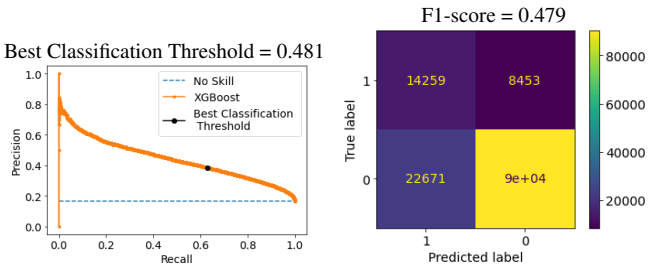
Here, we have taken *scale_pos_weight*= 6 based on the heuristic given in XGBoost documentation that suggests that a good value to try is the ratio of number of negative class to the positive class which in our case refers to count(non-diabetic)/count(diabetic). The hyperparameters used in the tuning process were

```
xgbc_param_space =
{'max_depth': hp.choice('max_depth', range(2,9)),
 'learning_rate': hp.uniform('learning_rate',0.05,0.5),
 'subsample': hp.uniform('subsample',0.5,1.0),
 'n_estimators': hp.choice('n_estimators', range(100,500)),
 'scale_pos_weight': hp.uniform('scale_pos_weight',3,6),
 'gamma': hp.uniform('gamma',0,0.3),
 'colsample_bytree': hp.uniform('colsample_bytree',0.5,1.0),
 'reg_lambda': hp.uniform('reg_lambda',1,4)}
```

The optimal hyperparamter values obtained for XGBoost:

```
{'colsample_bytree': 0.5, 'gamma': 0.2,
 'learning_rate': 0.067, 'max_depth': 2,
 'n_estimators': 200, 'reg_lambda': 2.947,
 'scale_pos_weight': 3.160, 'subsample': 0.593}
```

The best classification threshold value and the performance on the whole training set for the XGBoost model trained on optimal hyperparamter set (with predictions made based on the optimal threshold) is summarized in figure 9.



(a) Precision-Recall curve (b) Confusion matrix

Figure 9. XGBoost model trained on optimal hyperparamter set: (a) Precision-Recall curve with the the optimal threshold value and (b) Confusion Matrix (using optimal threshold) on the training data

5.4.4 LightGBM

Table 7 lists the mean F1-scores for 5-fold cross validation evaluated for different LightGBM models tried.

	F1-score
Default (max_depth = 3, num_leaves = 5)	0.253
scale_pos_weight = 6 (max_depth = 3, num_leaves = 5)	0.447
Optimal hyperparamters	0.476

Table 7. 5-fold cross validation mean F1-scores for LightGBM

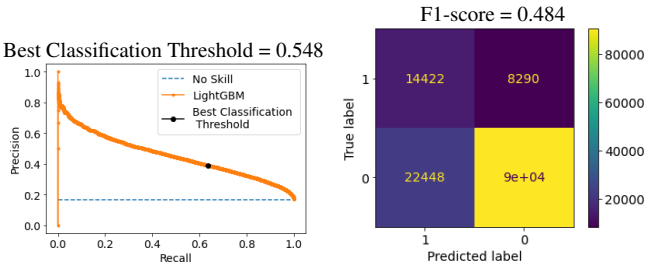
We use the same heuristic value for *scale_pos_weight* as done in XGBoost. Note that due to the leaf-wise strategy employed by LightGBM to grow trees, to avoid overfitting 'num_leaves' (maximum tree leaves) has to be less than 2^{max_depth} (LightGBM documentation). We took $num\_leaves = 0.6\left(2^{max\_depth}\right)$ for all LightGBM models tried. The hyperparameters used in the tuning process were

```
lgbmc_param_space =
{'max_depth': hp.choice('max_depth', range(3,9)),
 'learning_rate': hp.uniform('learning_rate',0.05,0.5),
 'subsample': hp.uniform('subsample',0.5,1.0),
 'n_estimators': hp.choice('n_estimators', range(100,500)),
 'scale_pos_weight': hp.uniform('scale_pos_weight',3,6),
 'min_split_gain': hp.uniform('min_split_gain',0,0.3),
 'colsample_bytree': hp.uniform('colsample_bytree',0.5,1.0),
 'reg_lambda': hp.uniform('reg_lambda',1,4)}
```

The optimal hyperparamter values obtained for LightGBM:

```
{'n_estimators': 491, 'learning_rate': 0.06,
 'max_depth': 3, 'colsample_bytree': 0.8,
 'min_split_gain': 0.246, 'reg_lambda': 3.203,
 'scale_pos_weight': 3.853, 'subsample': 0.528}
```

The best classification threshold value and the performance on the whole training set for the LightGBM model trained on optimal hyperparamter set (with predictions made based on the optimal threshold) is summarized in figure 10.



(a) Precision-Recall curve (b) Confusion matrix

Figure 10. LightGBM model trained on optimal hyperparamter set: (a) Precision-Recall curve with the the optimal threshold value and (b) Confusion Matrix (using optimal threshold) on the training data

5.4.5 CatBoost

Table 8 lists the mean F1-scores for 5-fold cross validation evaluated for different CatBoost models tried.

	F1-score
Default (n_estimators = 100)	0.264
scale_pos_weight = 6 (n_estimators = 100)	0.448
Optimal hyperparamters	0.478

Table 8. 5-fold cross validation mean F1-scores for CatBoost

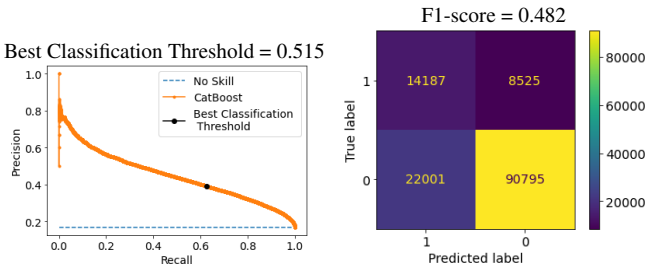
We use the same heuristic value for *scale_pos_weight* as done in XGBoost. The hyperparameters used in the tuning process were

```
catb_param_space =
{'max_depth': hp.choice('depth', range(3,9)),
 'learning_rate': hp.uniform('learning_rate',0.05,0.5),
 'subsample': hp.uniform('subsample',0.5,1.0),
 'n_estimators': hp.choice('n_estimators', range(100,500)),
 'scale_pos_weight': hp.uniform('scale_pos_weight',3,6),
 'rsm': hp.uniform('rsm',0.5,1.0),
 'reg_lambda': hp.uniform('reg_lambda',1,4)}
```

The optimal hyperparamter values obtained for CatBoost:

```
{'n_estimators': 170, 'learning_rate': 0.119,
 'max_depth': 4, 'reg_lambda': 3.248,
 'subsample': 0.559, 'rsm': 0.937,
 'scale_pos_weight': 3.341}
```

The best classification threshold value and the performance on the whole training set for the CatBoost model trained on optimal hyperparamter set (with predictions made based on the optimal threshold) is summarized in figure 11.



(a) Precision-Recall curve (b) Confusion matrix

Figure 11. CatBoost model trained on optimal hyperparamter set: (a) Precision-Recall curve with the the optimal threshold value and (b) Confusion Matrix (using optimal threshold) on the training data

5.4.6 Fully-connected neural network

We constructed a fully-connected neural network with 10 layers with different number of nodes. The structure was determined after several trials of different combination of layer depths and the number of nodes. The structure of the network is shown below in Figure 12.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1               [-1, 1, 33]             759
              ReLU-2               [-1, 1, 33]               0
            Linear-3               [-1, 1, 44]           1,496
              ReLU-4               [-1, 1, 44]               0
            Linear-5               [-1, 1, 55]           2,475
              ReLU-6               [-1, 1, 55]               0
            Linear-7               [-1, 1, 55]           3,080
              ReLU-8               [-1, 1, 55]               0
            Linear-9               [-1, 1, 55]           3,080
             ReLU-10               [-1, 1, 55]               0
           Linear-11               [-1, 1, 55]           3,080
             ReLU-12               [-1, 1, 55]               0
           Linear-13               [-1, 1, 44]           2,464
             ReLU-14               [-1, 1, 44]               0
           Linear-15               [-1, 1, 22]             990
             ReLU-16               [-1, 1, 22]               0
           Linear-17               [-1, 1, 11]             253
             ReLU-18               [-1, 1, 11]               0
           Linear-19                [-1, 1, 2]              24
================================================================
```

Figure 12. Structure of the fully-connected neural network.

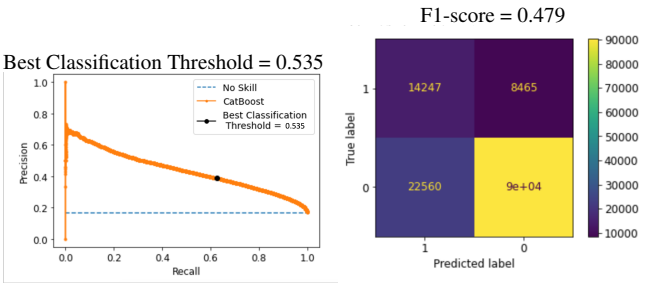
The options of hyperparameters tuned are shown here. Note that due to runtime constraints, only some but not all combinations of hyperparameters shown below were trained and tested, and there are 26 trained networks in total.

```
param_grid =
{'echos': [15, 30],
 'batch_size': [32, 128, 512],
 'learning rate': [0.01, 0.001],
 'weight': [3.5, 3.8, 4.15, 4.5],
 'optimizer': ['SGD', 'Adagrad']}
```

The optimal hyperparamter values obtained for FCNN:

```
{'epochs': 15, 'batch_size': 32,
 'learning_rate': 0.01, 'weight': 4.15,
 'optimizer': 'Adagrad'}
```

The best classification threshold value and the performance on the whole training set for the neural network model trained on optimal hyperparamter set (with predictions made based on the optimal threshold) is summarized in figure 13.



(a) Precision-Recall curve (b) Confusion matrix

Figure 13. FCNN model trained on optimal hyperparamter set: (a) Precision-Recall curve with the the optimal threshold value and (b) Confusion Matrix (using optimal threshold) on the training data

5.4.7 Undersampling the training data

We explored the strategy of tackling class imbalance using undersampling by training some of the classification models discussed above on an undersampled training set. The majority class (non-diabetic) in the imbalanced training set was randomly sampled using imblearn.under_sampling.RandomUnderSampler such that the under-sampled training set has equal number of samples from both classes. We followed the similar analysis pipeline as before where we try the model with default parameters and then tune the hyperparameters. Considering that the model is now trained on 'balanced' training set, we did not do optimal threshold calculation for these cases. Their performance is compared with the other models trained on the whole imbalanced training data using the F1-score on the test set in the next section.

5.4.8 Performance on the test set

Table 9 lists the F1-score, precision and recall on the test set for the binary classification models trained on their respective optimal hyperparameter sets. Class predictions on the test set are made using their respective optimal classification threshold (except in baseline and undersampled training data cases where default threshold of 0.5 is used).

Model	F1-score	Recall	Precision
Baseline			
Vanilla Logistic Regression	0.248	0.162	0.537
Asymmetrical cost function, and optimal classification threshold			
Logistic Regression	0.468	0.655	0.364
Random Forest	0.471	0.617	0.381
XGBoost	0.477	0.629	0.384
LightGBM	**0.481**	**0.624**	**0.391**
CatBoost	0.478	0.613	0.391
Neural Net	0.475	0.616	0.387
Fit to undersampled training data			
Logistic Regression	0.461	0.723	0.338
Random Forest	0.457	0.763	0.326
LightGBM	0.463	0.761	0.332

Table 9. Scores on the test set obtained using the model specific optimal classification threshold values

Comparing the performance of two strategies of handling data imbalance, we can see the performance of combined strategy of asymmetric cost function and optimal classification threshold performs better than using undersampled training data.

Furthermore, among all the models considered, LightGBM model trained with asymmetric cost function along with using optimal threshold for classification has the highest F1-score on the test set. Hence, we selected this LightGBM based binary classification model for further analysis.

5.5. Performance comparison across Years

Since the training and testing for models was done on only the cleaned 2015 dataset, we were interested in seeing how the selected model performs on the datasets from other years. Table 10 summarizes the performance of the selected LightGBM classification model on the cleaned datasets of other years (2011 - 2014). We observed that the F1-score, precision and recall values for all the five years are similar. This lends support to the assumption in problem formulation (section 3) of taking cleaned 2015 dataset as the representative of all the five year data

Year	F1-score	Recall	Precision
2011	0.463	0.608	0.374
2012	0.465	0.611	0.375
2013	0.471	0.618	0.380
2014	0.474	0.622	0.383
2015	0.483	0.633	0.391

Table 10. Performance of LightGBM classifier on years 2011-2015

5.6. Model Interpretation using SHAP

In order to explain how different features contribute to prediction capabilities of the selected LightGBM model, we used game theory based SHAP (SHapley Additive exPlanations) methodology for model interpretation [7] (Package: shap). The SHAP summary plot in figure 14 shows that the 6 most features for cleaned 2015 dataset in decreasing order are GENHLTH, _BMI5, _AGE5YR, CHECKUP, SEX and INCOME2. The mean(|SHAP value|) for these features was in the range 0.14 - 0.6, on the other hand the mean value for each of the remaining 16 features is ≤ 0.05. This indicates that the combined impact of top 6 features on the model output is much higher than the total impact of the remaining 16 features.

We also performed SHAP on the cleaned datasets of other years (figure 15). We found that top 6 features for all the four years were same as those for the cleaned 2015 dataset as shown in figure 15. The order of features was also same as that observed for 2015, except for 2^{nd} and 3^{rd} most important features being swapped for years 2011 and 2012.

The top 6 features are described in table 11. The SHAP summary plots show that high values of GENHLTH push the model prediction towards *diabetic* class. This suggests that respondents who consider their general health to be 'poor' or 'fair' are more likely to be diabetic than who answer 'excellent' or 'very good'. Similar to GENHLTH, higher values of features _BMI5 and _AGE5YR also push class prediction towards 'diabetic'. This is consistent with what we expect based on literature, that is older people or people having weight status of 'overweight' or 'obese' (figure 3 are more prone to diabetes. From a socioeconomics point of view, we observe that lower household income pushes the prediction to 'diabetic'. Furthermore, interestingly, the SHAP values suggest that males are more likely to be diabetic than females.

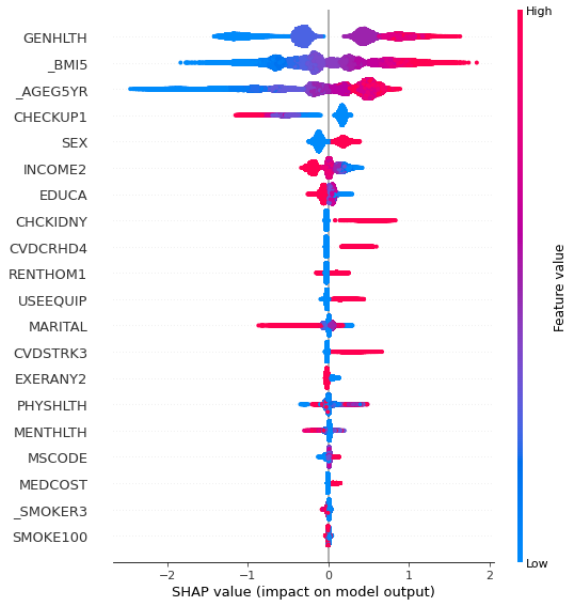
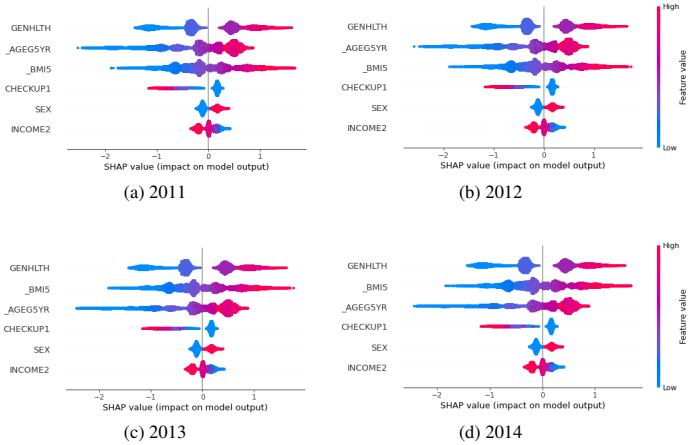


Figure 14. SHAP summary plot of the *diabetic* class for the selected LightGBM model on 2015 dataset

GENHLTH	Would you say that in general your health is? 1: Excellent, 2: Very Good, 3: Good, 4: Fair, 5: Poor
_BMI5	Body Mass Index Real valued
_AGE5YR	Fourteen-level age category 1: 18-24, 2: 25-29, 3: 30-34 , ... , 12: 75-79, 13: $\geq$ 80
CHECKUP1	About how long has it been since you last visited a doctor for a routine checkup? 0: Never, 1: within past year, 2: within past 2 years, 3: within past 5 years, 4: 5 or more years
SEX	Indicate sex of respondent 0: Female, 1: Male
INCOME2	Is your annual household income from all sources? 1: $<$ 10k, 2: 10k-15k, 3: 15k-20k, 4: 20k-25k, 5: 25k-35k, 6: 35k-50k, 7: 50k-75k, 8: $>$ 75k

Table 11. Top 6 features across the year based on SHAP for the selected LightGBM model



(a) 2011 (b) 2012

(c) 2013 (d) 2014

Figure 15. SHAP summary plots with top 6 features for the selected LightGBM model on 2011-2014 datasets

6. Discussion

In this study, we train binary classifier models using the CDC BRFSS data, aiming to predict respondents that could have type II diabetes. We performed manual feature selection and data clean-up following previous work by Xie et al. [11] and Alex Teboul [9]. Due to class imbalance between *diabetic* and *non-diabetic*, our model aims to achieve maximization of precision and recall instead of accuracy, and therefore we measured our models performance with the F1 score. We tackled class imbalance by three means, altering the class weight in the *logloss* function (eqn. 2), choosing optimal threshold for F1 score, and performing under-sampling. We chose the symmetric logistic regression as our baseline model, and other models we analyzed include asymmetric logistic regression, random forest, neural network, gradient boosting (XGBoost, LightGBM, and CatBoost). We would also like to note that although we had wanted to test support vector machines (SVMs) as well, the large number of observations in our dataset made SVM an impractical choice due to very high computation time. Despite the long run time, a single run using a linear kernel, no regularization, and "balanced" class weights did not have better performance than other models.

We trained all models on the cleaned 2015 dataset, which was split into train and test sets that maintained the original class imbalance (table 3). We tackled class imbalance using a two-fold approach of altering the class weight in the *logloss* function (eqn. 2) and choosing optimal classification threshold. In all the models, we tuned the class weights and calculated the optimized threshold using the precision-recall curve. We observed that **altering class weight improved model performance notably**, and optimizing the threshold further improved performance marginally. Of all the models we compared, **LightGBM was the best-performing** with the highest F1-score on the test set, and we showed that this model had comparable performance on the data from 2011 to 2014.

We also explored the method of handling class imbalance by undersampling the majority class in training set such that there is equal proportion of diabetic and non-diabetic classes. We trained logistic regression, random forest and LightGBM classifiers on undersampled training data and evaluated these models' performance on the test set. We found that undersampling did not improve performance more than the combined strategy of asymmetric cost and optimal threshold (table 3).

Finally, we analyzed the feature importance of our best model, LightGBM, and concluded that the top six most important features were general health, BMI, age, checkup interval, sex, and income. The SHAP values showed that **these features together contributed much more to the model output than the other 16 features combined.** The same analysis on datasets of other years suggested the similar importance of the top 6 features, only the order of age and BMI being swapped in 2011 and 2012.

References

[1] Jason Brownlee. *A Gentle Introduction to Threshold-Moving for Imbalanced Classification*. URL: https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/.

[2] Jason Brownlee. *A Gentle Introduction to XGBoost for Applied Machine Learning*. URL: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/.

[3] Centers for Disease Control and Prevention. *Annual Survey Data*. URL: https://www.cdc.gov/brfss/annual_data/annual_data.htm.

[4] Centers for Disease Control and Prevention. *Behavioral Risk Factor Surveillance System*. URL: https://www.kaggle.com/cdc/behavioral-risk-factor-surveillance-system.

[5] Centers for Disease Control and Prevention. *National Diabetes Statistics Report, 2020*. URL: https://www.cdc.gov/diabetes/data/statistics-report/index.html.

[6] Bartosz Krawczyk. "Learning from imbalanced data: open challenges and future directions". In: *Progress in Artificial Intelligence* 5.4 (2016), pp. 221–232.

[7] Christoph Molnar. *Interpretable Machine Learning*. URL: https://christophm.github.io/interpretable-ml-book/shap.html.

[8] Baptiste Rocca. *Handling imbalanced datasets in machine learning*. URL: https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28.

[9] Alex Teboul. *Diabetes Health Indicators Dataset*. URL: https://www.kaggle.com/alexteboul/diabetes-health-indicators-dataset.

[10] Kris Wright. *XGBoost, LightGBM, and Other Kaggle Competition Favorites*. URL: https://districtdatalabs.silvrback.com/parameter-tuning-with-hyperopt.

[11] Zidian Xie. "Building Risk Prediction Models for Type 2 Diabetes Using Machine Learning Techniques". In: *Preventing Chronic Disease* 16 (2019). URL: http://dx.doi.org/10.5888/pcd16.190109.

[12] Andre Ye. *XGBoost, LightGBM, and Other Kaggle Competition Favorites*. URL: https://towardsdatascience.com/xgboost-lightgbm-and-other-kaggle-competition-favorites-6212e8b0e835.