REPORT ON THE IMPLEMENTATION OF THE BEST CLASSIFICATION MODEL FOR DIABETES PREDICTION

Principles of Data Mining and Machine Learning (2022 MOD007892)

SID: 2158452

TABLE OF CONTENTS

TABLE OF CONTENTS 1

LIST OF FIGURES 2

INTRODUCTION 3

DATA UNDERSTANDING 4

EXPLORATORY DATA ANALYSIS 4

DATA PREPARATION 5

OUTLIERS REMOVAL 5

MISSING VALUES 5

FEATURE ENGINEERING 5

CLASS IMBALANCE 5

FEATURE SCALING 6

FEATURE ENCODING 6

MODELING 7

MODELS 7

WORKFLOW 7

CROSS VALIDATION & HYPER PARAMETER TUNING 7

EVALUATION 8

CONFUSION MATRIX 8

EVALUATION METRICS 8

COMPARISON 9

CONCLUSION 9

LIST OF FIGURES

Figure 1: Description of Variables in the Diabetes Dataset 4
Figure 2: Summary Statistics of the Variables with Emphasis on Missing Values and Outliers 4
Figure 3: A bar chart showing the class imbalance in the Outcome variable 5
Figure 4: Explanation of Each Value in the Confusion Matrix 8
Figure 5: Explanation and Formula for Evaluation Metrics 8
Figure 6: Comparison Dataframe Showing the Evaluation Metrics of Each Model 9
Figure 7: Confusion Matrix for the Logistic Regression Model 10
Figure 8: Logistic Regression Hyperparameters and Explanation 10
Figure 9: Confusion Matrix for KNN Model 10
Figure 10: KNN Hyperparameters and their Explanation 11
Figure 11: Confusion Matrix for SVM Model 11
Figure 12: SVM Hyperparameters and their Explanation 11
Figure 13: Confusion Matrix for the Decision Tree Model 12
Figure 14: Decision Tree Hyperparameters and their Explanation 12
Figure 15: Confusion Matrix for the Random Forest Model 12
Figure 16: Random Forest Hyperparameters and their Explanation 13
Figure 17: Random Forest Classifier's Ranking of Features in the Dataset according to their Importance 13
Figure 18: Confusion Matrix for the Naive Bayes Model 13

INTRODUCTION

Diabetes is a growing concern worldwide, with over 400 million people affected globally. The disease is associated with numerous complications, including cardiovascular disease, kidney disease, and nerve damage. As such, early detection and prevention is crucial.

The Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology was used for this diabetes prediction project. The methodology was modified to the following 4 stages for the project.

1. Data Understanding
2. Data Preparation
3. Modeling
4. Evaluation

DATA UNDERSTANDING

The dataset contains information indicated in the figure below for 768 individuals.

Feature	Description
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
BloodPressure	Diastolic blood pressure (mm Hg)
SkinThickness	Triceps skin fold thickness (mm)
Insulin	2-Hour serum insulin (mu U/ml)
BMI	Body mass index (weight in kg/(height in m)^2)
DiabetesPedigreeFunction	Diabetes pedigree function
Age	Age (years)
Outcome	Class variable (0 or 1)

Figure 1: Description of Variables in the Diabetes Dataset

EXPLORATORY DATA ANALYSIS

The summary statistics revealed that some variables had 0 as the minimum value, which should not be possible (Fig. 2). This probably meant that missing values in the dataset might have been encoded as 0.

Histograms, box plots, violin plots, and correlation charts of variables were then explored to check for outliers and generate ideas for feature engineering.

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

Figure 2: Summary Statistics of the Variables with Emphasis on Missing Values and Outliers

DATA PREPARATION

OUTLIERS REMOVAL

A function that standardizes all values in each variable and removes any observations containing values greater than or equal to 3 (the threshold for outliers) in any of the columns was applied which pruned the dataset to 719 records.

MISSING VALUES

Missing values encoded as zeros were imputed with the mean value of the respective variable.

FEATURE ENGINEERING

Three variables were created by binning the Glucose, Age, and BMI variables. Two other variables were created from the interaction between Pregnancy, Glucose and BMI variables.

CLASS IMBALANCE

The data set was found to be imbalanced as the negatives (66%) are more than the positives (34%) and there is a high likelihood that the classifier overfits the negative class. This was addressed with the Synthetic Minority Over-sampling Technique (SMOTE) oversampling technique which upsampled the dataset to create a balance between the classes.

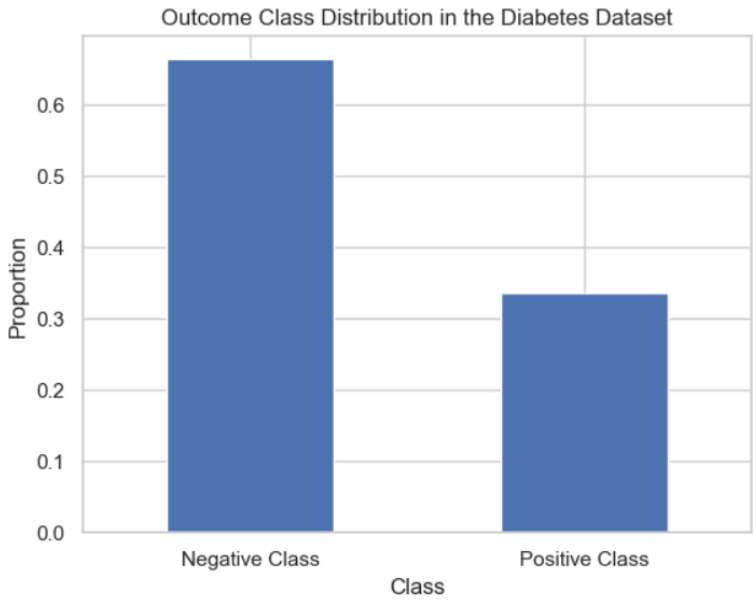

Figure 3: A bar chart showing the class imbalance in the Outcome variable

FEATURE SCALING

The StandardScaler function was used for scaling all the numeric variables. This ensures that they are on the same scale and aims to prevent the models from assigning more weights to variables with higher values.

FEATURE ENCODING

The three newly created categorical variables are ordinal variables and were therefore encoded with the OrdinalEncoder function before being fed to the model.

MODELING

MODELS

Since the dataset has a label which is categorical, the task required a supervised classification algorithm, and the following models were implemented: Logistic Regression, K-Nearest Neighbors (K-NN), Support Vector Machine, Decision Tree. Random Forest, and Naïve Bayes.

WORKFLOW

The dataset was split into train and test sets in the 80/20 ratio. A copy was then made for each model to ensure impartial evaluation. Also, the same random state seed was used for all operations to ensure reproducibility. A column transformer instance that standardizes numeric variables and scale ordinal variables was placed into a pipeline that sequentially transforms the variables and runs the estimator (model). The pipeline was then used as part of the parameters for the GridSearchCV function.

CROSS VALIDATION & HYPER PARAMETER TUNING

The hyperparameters of each model were tuned except the Naïve Bayes classifier which does not have any hyperparameter to tune. The best parameters were found using GridSearchCV except for Decision Tree, Support Vector Machine and Random Forest Classifiers where RandomizedSearchCV was done instead as GridSearchCV would be too computationally expensive. The best estimator for each model was then used to predict the values of the label. The different hyperparameters tuned for each model and the explanation can be found in the Appendix.

EVALUATION

CONFUSION MATRIX

The confusion matrix is a table used to evaluate the performance of a binary classification algorithm, showing the predicted and actual target values in a cross-tabulated format. The sum of all the values in the matrix equals the number of records in the test set which is 144 in this case. The confusion matrices of all the models can be found in the Appendix. The figure below explains each value in the matrix.

> The Upper Left values represent the Negatives correctly identified: True Negatives
>
> The Lower Left values represent the Positives incorrectly identified as Negatives: False Negatives
>
> The Upper Right values represent the Negatives incorrectly identified as Positives: False Positives
>
> The Lower Right values represent the Positives correctly identified as Positives: True Positives

Figure 4: Explanation of Each Value in the Confusion Matrix

EVALUATION METRICS

The metrics except the AUROC used for evaluating and comparing the implemented models are extracted from the confusion matrix of each model.

Metric	Description
Accuracy	Proportion of correct predictions (True Positives and True Negatives) made by each classifier. Formula: `(TP + TN) / (TP + TN + FP + FN)`
Precision	Proportion of actual positive instances that are correctly identified by a classifier. Formula: `TP / (TP + FP)`
Recall/Sensitivity/TPR	Proportion of actual positive instances (True Positive) identified correctly by each classifier. Formula: `TP / (TP + FN)`
Specificity/TNR	Proportion of actual negative instances (True Negative) identified correctly by each classifier. Formula: `TN / (TN + FP)`
F1 Score	The harmonic mean of precision and recall scores, which provides a balance between these two metrics. It is a way to measure the effectiveness of a classifier at both identifying positive instances and avoiding false positives. Formula: `2 * Precision * Recall / (Precision + Recall)`
Area Under the ROC Curve	Measures the ability of a classifier to distinguish between positive and negative instances.

Figure 5: Explanation and Formula for Evaluation Metrics

COMPARISON

The metrics for all the models were converted to percentages and compiled into a dataframe for easier readability and comparison.

Out[79]:

	Model	Specificity (%)	Recall Score/Sensitivity (%)	AUROC Score (%)	F1 Score (%)	Accuracy Score (%)	Precision Score (%)
1	Logistic Regression	87.50	66.67	88.24	69.57	80.56	72.73
2	KNN	84.38	70.83	85.35	70.10	79.86	69.39
3	Support Vector Machine	84.38	68.75	83.90	68.75	79.17	68.75
4	Decision Tree	86.46	50.00	81.49	56.47	74.31	64.86
5	Random Forest	85.42	79.17	90.17	76.00	83.33	73.08
6	Naive Bayes	73.96	79.17	88.37	68.47	75.69	60.32

Figure 6: Comparison Dataframe Showing the Evaluation Metrics of Each Model

We can see from the output that all the classifiers still tend to be accurate with their prediction of negative class than the positive class which might be due to the class imbalance. That is where the Area under the Receiver Operating Characteristic (ROC) Curve comes in. This metric measures the ability of a classifier to distinguish between positive and negative instances and is particularly useful when the classes are imbalanced. It is a way to evaluate the overall performance of a classifier, as it considers both true positive rate (Recall/Sensitivity) and false positive rate (Specificity).

CONCLUSION

On this basis and from the evaluation dataframe (Fig. 6), **Random Forest Classifier** is the best classifier as it has the highest AUROC metric, F1 metric, accuracy metric and sensitivity while maintaining a high specificity: That means it stands a better chance of accurately predicting whether some would have or not have diabetes within the next five years than other models.

APPENDIX

1. Logistic Regression

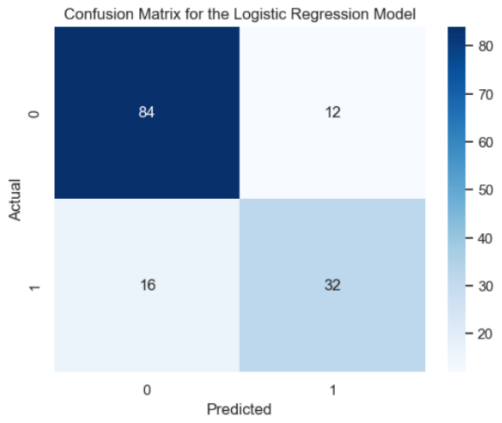

Figure 7: Confusion Matrix for the Logistic Regression Model

Parameter	Explanation
logit__penalty	Type of regularization used by the logistic regression model. `l1` corresponds to L1 regularization (lasso), which can lead to sparse models, and `l2` corresponds to L2 regularization (ridge), which typically produces models with smaller coefficients.
logit__C	The inverse of the regularization strength. Smaller values of `C` correspond to stronger regularization, which can reduce overfitting.
logit__class_weight	Weighting strategy for the classes in the dataset. `balanced` adjusts the weights inversely proportional to the class frequencies, which can help to handle imbalanced datasets. `None` means that no weights are applied.
logit__solver	Algorithm used to optimize the logistic regression objective function. `liblinear` is a coordinate descent solver that works well for smaller datasets, while `saga` is a stochastic gradient descent solver that is often faster for larger datasets.
logit__max_iter	Maximum number of iterations taken for the solvers to converge.
logit__fit_intercept	Whether to calculate the intercept for this model. Set to `False` if the data is already centered.

Figure 8: Logistic Regression Hyperparameters and Explanation

2. KNN

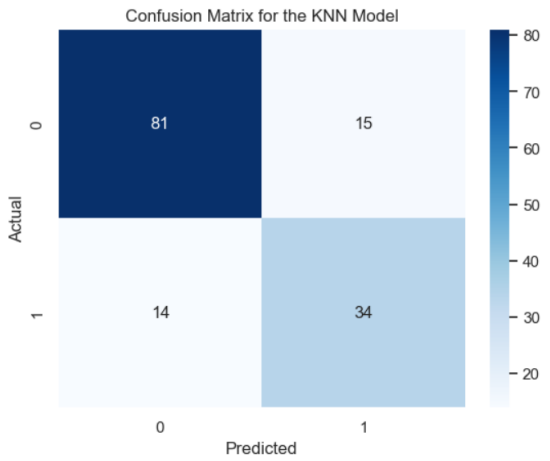

Figure 9: Confusion Matrix for KNN Model

Parameter	Explanation
knn__n_neighbors	Number of neighbors to use for classification. Larger values can result in smoother decision boundaries, while smaller values can lead to more complex models.
knn__weights	Weight function used in prediction. `uniform` assigns equal weight to each neighbor, while `distance` assigns weights proportional to the inverse of the distance from the query point.
knn__algorithm	Algorithm used to compute the nearest neighbors. `auto` selects the best algorithm based on the training data, while `ball_tree` and `kd_tree` use tree structures to speed up the computation, and `brute` performs a brute-force search over all points.
knn__p	Power parameter for the Minkowski metric. When `p=1`, this corresponds to the Manhattan distance, and when `p=2`, this corresponds to the Euclidean distance. Other values of `p` can be used to compute other distance metrics.

Figure 10: KNN Hyperparameters and their Explanation

3. SVM

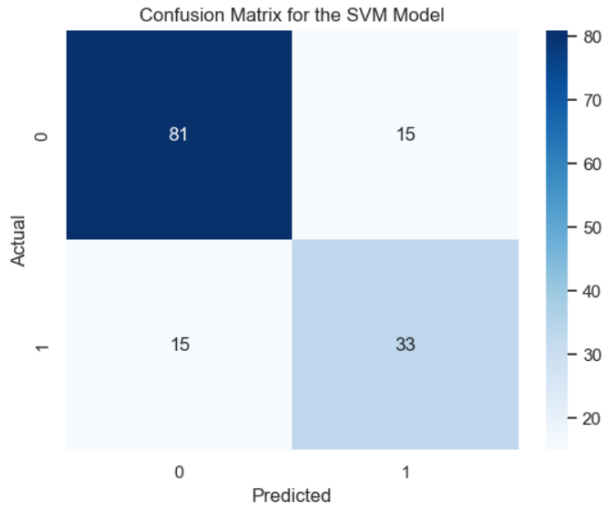

Figure 11: Confusion Matrix for SVM Model

Parameter	Explanation
svm__kernel	Specifies the kernel type to be used in the algorithm. `linear` works well for linearly separable data, `poly` can work well for non-linearly separable data by mapping the data into a higher-dimensional space, and `rbf` can also work well for non-linearly separable data by mapping the data into a higher-dimensional space using radial basis functions.
svm__C	Regularization parameter. Larger values of `C` correspond to less regularization, resulting in more complex models that are better able to fit the training data. Smaller values of `C` correspond to more regularization, which can help prevent overfitting.
svm__degree	Degree of the polynomial kernel function. This parameter is only used when the `poly` kernel is selected. Larger values of `degree` correspond to more complex models, but can also result in overfitting.
svm__gamma	Kernel coefficient for the `rbf` kernel function. Larger values of `gamma` correspond to more complex models, and can result in overfitting. Smaller values of `gamma` can help prevent overfitting by limiting the influence of individual training examples.

Figure 12: SVM Hyperparameters and their Explanation

4. Decision Tree

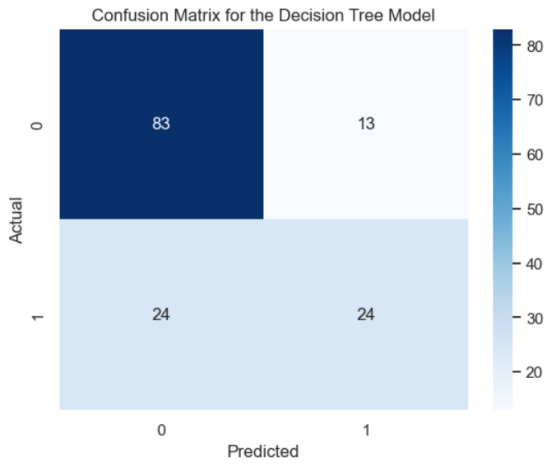

Figure 13: Confusion Matrix for the Decision Tree Model

Parameter	Explanation
`dt__criterion`	Criterion to measure the quality of split. `gini` measures the impurity of a node based on the probability of incorrectly classifying a randomly chosen element in the node, while `entropy` measures the impurity of a node based on the information gain from splitting the node.
`dt__max_depth`	Maximum depth of the tree. If `None`, nodes are expanded until all leaves are pure or until all leaves contain less than `min_samples_split` samples.
`dt__min_samples_split`	Minimum number of samples required to split an internal node. If fewer than `min_samples_split` samples are at a node, the node will not be split.
`dt__min_samples_leaf`	Minimum number of samples required to be at a leaf node. If a split would result in fewer than `min_samples_leaf` samples in a leaf, the split is ignored.
`dt__max_features`	The number of features to consider when looking for the best split. `auto` considers all features, `sqrt` considers the square root of the number of features, and `log2` considers the logarithm base 2 of the number of features.
`dt__class_weight`	Weights associated with classes in the input data. If `None`, all classes are supposed to have weight one. `balanced` mode adjusts the weights to be inversely proportional to class frequencies in the input data.

Figure 14: Decision Tree Hyperparameters and their Explanation

5. Random Forest

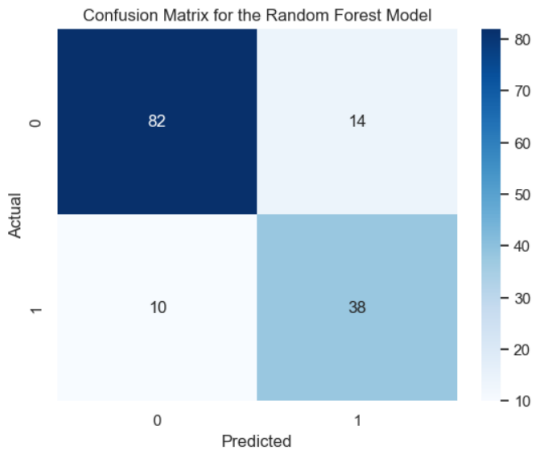

Figure 15: Confusion Matrix for the Random Forest Model

Parameter	Explanation
rfc__n_estimators	The number of trees in the forest
rfc__max_depth	Maximum depth of the tree
rfc__max_features	The number of features to consider when looking for the best split
rfc__min_samples_split	Minimum number of samples required to split an internal node
rfc__min_samples_leaf	Minimum number of samples required to be at a leaf node

Figure 16: Random Forest Hyperparameters and their Explanation

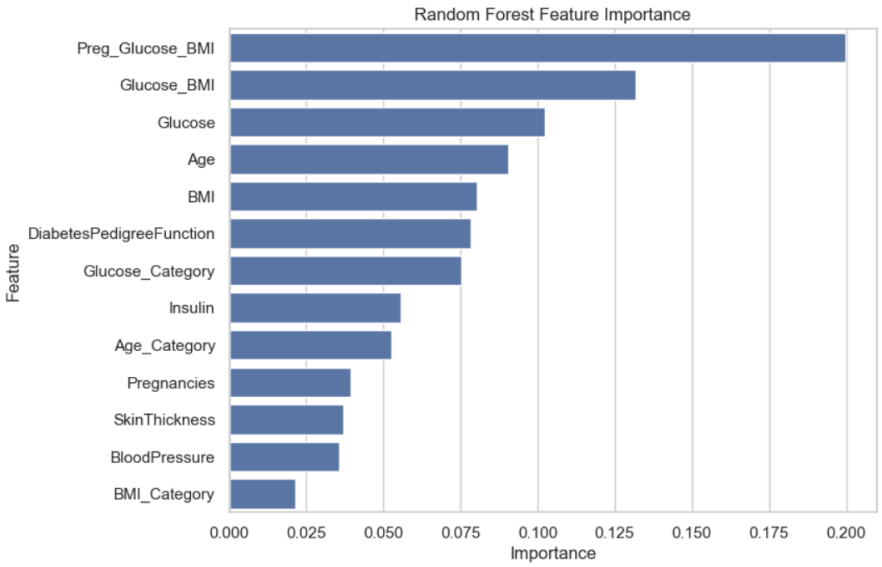

Figure 17: Random Forest Classifier's Ranking of Features in the Dataset according to their Importance

6. Naïve Bayes

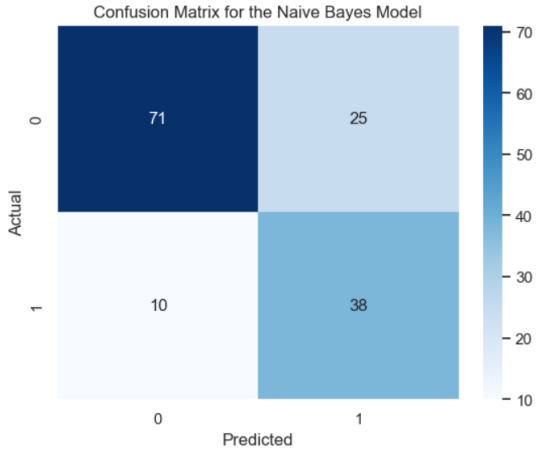

Figure 18: Confusion Matrix for the Naive Bayes Model