Diabetes Prediction Using Machine Learning

DSCI3415 Project Phase V

Model & Utility Application Implementation

Mona Ibrahim
Mathematics and Actuarial Science
American University in Cairo
900212749
monamahmoud@aucegypt.edu

Maya Elshweikhy
Mathematics and Actuarial Science
American University in Cairo
900204233
mayahanyy1@aucegypt.edu

In this report, we outline the final model design for predicting diabetes using Support Vector Machines (SVMs). This model selection follows a thorough analysis of model implementation choices, model design, including architecture and model training and validation, model performance, and utility application implementation.

1. Model Implementation Choices

After testing different models in the previous phase using *sklearn*: Logistic regression, Decision Tree, Random Forest, Naive Bayes, SVM, AdaBoostClassifier, KNN, Gradient Boosting, and XgBoost. Then, the model performance was tested for each model. The metrics of accuracy, precision, recall, and F1 score were used to evaluate the performance of each model. The top three models were: Support Vector Machine, Random Forest, and Decision Tree.

Models' Performance Evaluation. Precision, recall, and F1 scores were calculated for each model using all observations (both groups).

Model	Accuracy	Precision	Recall	F1
SVM	**0.94**	**0.97**	**0.88**	**0.92**
Random Forest	**0.91**	**0.93**	**0.85**	**0.89**
Decision Tree	**0.89**	**0.90**	**0.83**	**0.86**

The SVM model had best performance as shown in table above using *sklearn*. Therefore, we chose to implement SVM as it will be able to classify individuals into either diabatic or non-diabatic classes. It has higher accuracy, precision and recall and F1 compared to other models. The SVM model exhibited a good model performance which makes it suitable for the prediction.

2. Model Design

General Architecture

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression analysis. As shown in Figure 1, SVM finds the optimal hyperplane for separating classes linearly by finding the largest margin, the minimum distance from perpendicular distances from each observation point to the separating hyperplane through kernel trick.

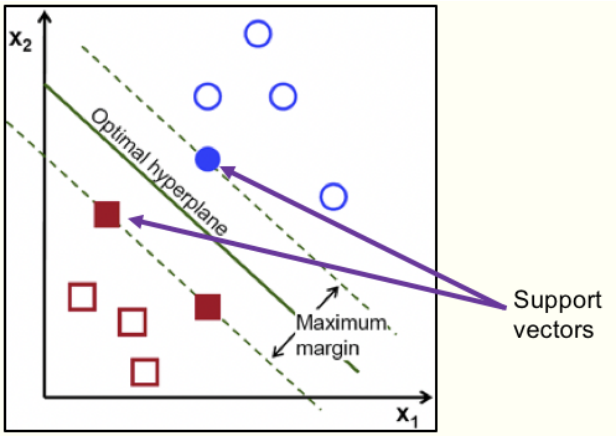

Figure 1. Support Vector Machines (SVM) Classification Plot.

The SVMs have hyperparameters that need to be tuned for optimal performance, such as the choice of kernel function, regularization parameter (C), and kernel-specific parameters (e.g., gamma for radial basis function kernel). As shown in Figure 2, the different kernel functions along with their formula and optimization parameter(s). These kernel functions are: "linear or dot product", "rbf", "poly", and "sigmoid". In

SVM the default parameter is “rbf”. The “rbf” (radial basis function) and “poly”(polynomial kernel) are useful for non-linear hyper-plane and are called nonlinear svm.

No	Kernel function	Formula	Optimization parameter
1	Dot -product	$K(x_n,x_i)=(x_n,x_i)$	C
2	RBF	$K(x_n,x_i)=\exp(-\gamma \parallel x_n-x_i \parallel^2+C$	C and γ
3	Sigmoid	$K(x_n,x_i)=\tanh(\gamma(x_n,x_i)+r)$	C, γ, and r
4	Poly-nomial	$K(x_n,x_i)=(\gamma(x_n,x_i)+r)^d$	C, γ, r, d

Figure 2. Support Vector Machines (SVM) Kernel Functions.

Training the Model

SVM fits with our dataset problem as SVMs are sensitive to the scale of features, and we were able to successfully deal with that. Since we did proper feature scaling (normalization), we ensured that features were on a similar scale, preventing features with larger magnitudes from dominating the decision function.

Parameters used: the regularization parameter C was set to 1 (as the default value). The rbf kernel was used. Then we calculated the accuracy score, and it turned out to be 0.79.

```
Train Set Accuracy:83.55048859934854
Test Set Accuracy:78.57142857142857
```

Figure 3. Accuracy Score of SVM Initial Model.

And so we did Grid search hyperparameter optimization to find the best combination of hyperparameters. Additionally, SVM's margin maximization approach makes it less prone to overfitting compared to Decision Trees and Random Forests, which might be crucial in medical diagnosis tasks where generalization is essential. Lastly, SVMs can handle small to medium-sized datasets efficiently, like our Pima Indian dataset, without requiring extensive computational resources. Since we did proper preprocessing and model tuning and compared its performance scores, the SVM should produce the best results in predicting diabetes.

Hyperparameter Optimization

We wanted to improve the score and avoid overfitting, so as shown in Figure 4, the Grid Search CV algorithm was used and the SVM classifier was repeated to observe the new accuracy score. We used the function *RepeatedStratifiedKFold* which is a cross-validator that provides train/test indices to split data into train/test sets. It repeats Stratified K-Fold cross-validation *n_repeats* times with *n_splits* splits. Stratification ensures that each fold has the same proportion of target classes as the entire dataset, which is crucial for maintaining the distribution of classes across folds. The *random_state* parameter sets the seed for reproducibility.

```
# define grid search
grid = dict(kernel=kernel,C=C,gamma=gamma)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv,
                           scoring='f1',error_score=0)
```

Figure 4. Grid Search CV Algorithm of SVM.

Parameters chosen by Grid Search CV that had best performance: the regularization parameter C was set to 50 , gamma is set to scale, and kernel is rbf. The new accuracy score is 0.94, which shows that the Grid Search algorithm significantly affected the accuracy score.

```
Train Set Accuracy:91.69381107491856
Test Set Accuracy:93.5064935064935
```

Figure 4. Accuracy Score of SVM Using Grid Search CV.

```
Classification Report is:
              precision    recall  f1-score   support

           0       0.92      0.98      0.95        89
           1       0.97      0.88      0.92        65

    accuracy                           0.94       154
   macro avg       0.94      0.93      0.93       154
weighted avg       0.94      0.94      0.93       154
```

Figure 5. Classification report of SVM.

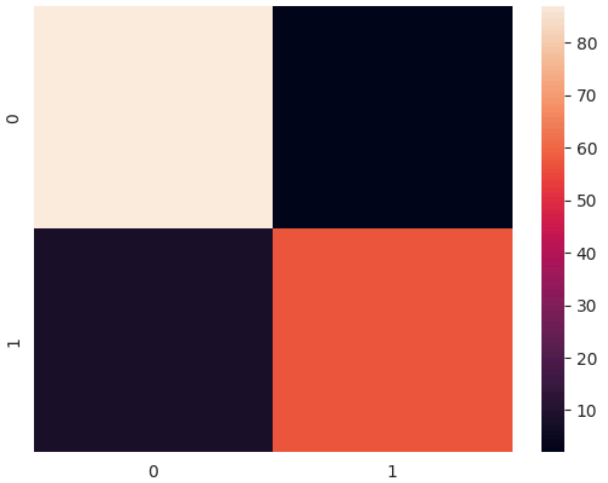

Figure 6. Confusion matrix of SVM.

Pros:

1. Robust against overfitting: SVMs are less prone to overfitting, especially in high-dimensional spaces, due to the margin maximization principle. They tend to generalize well to unseen data.
2. Versatility in kernel selection: SVMs can use different kernel functions (e.g., linear, polynomial, radial basis function) to capture complex relationships between features and the target variable. This flexibility allows SVMs to model non-linear relationships effectively.
3. Works well with small to medium-sized datasets: SVMs can handle small to medium-sized datasets efficiently, including the Pima Indian dataset, without requiring extensive computational resources.

Cons:

1. Computationally intensive: Training SVMs can be computationally intensive, especially for large datasets, non-linear kernels, or high-dimensional feature spaces. It may require more time and computational resources compared to other algorithms like logistic regression or decision trees.
2. Limited interpretability: SVMs provide little insight into the underlying decision-making process, making them less interpretable compared to simpler models like logistic regression or decision trees.

Overall, SVM can be effective for binary classification tasks like predicting diabetes in the Pima Indian dataset, especially when the dataset is not extremely large and when the choice of kernel and hyperparameters is carefully optimized (like we did using Grid Search). However, they require careful tuning and may be less interpretable compared to some other algorithms.

3. Utility Application Choice

To implement a utility application, we will be using the Python library PyQt5 to interact with the user by asking questions to collect the input data. The application is interactive, so it asks the user to give an input. Also, errors were handled in which if the input feature is outside of range, it will validate the user input to be within range before proceeding with the following input feature.

To use the PyQt5 library, we created a .py file that implements the SVM on our PIMA dataset and saves the model using the pickle library. Then we created another notebook that has the GUI application and connected it to the py file that contains the model. The GUI code implements windows with buttons and text as shown below.

```
df_feature_imp.describe()
```

	Pregnancies	Glucose	SkinThickness	Insulin	Age	BMIxThickness	BMIxAge	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000
mean	3.845052	121.677083	29.089844	141.753906	33.240885	978.102734	1080.40651	0.348958
std	3.369578	30.464161	8.890820	89.100847	11.760232	453.893312	437.98205	0.476951
min	0.000000	44.000000	7.000000	14.000000	21.000000	137.200000	382.20000	0.000000
25%	1.000000	99.750000	25.000000	102.500000	24.000000	664.575000	743.32500	0.000000
50%	3.000000	117.000000	28.000000	102.500000	29.000000	923.700000	989.35000	0.000000
75%	6.000000	140.250000	32.000000	169.500000	41.000000	1202.725000	1357.20000	1.000000
max	17.000000	199.000000	99.000000	846.000000	81.000000	3742.200000	2697.00000	1.000000

Figure 7. Descriptive Statistics of the Data

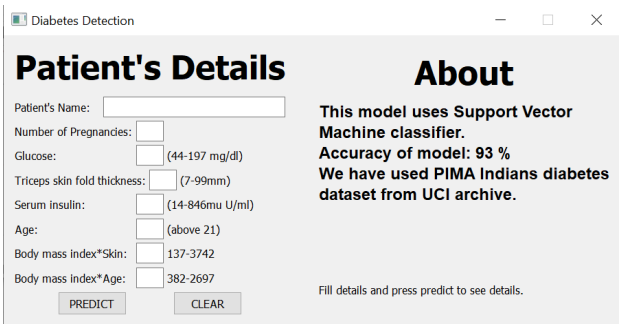

Figure 8. Diabetes Detection Application Window.

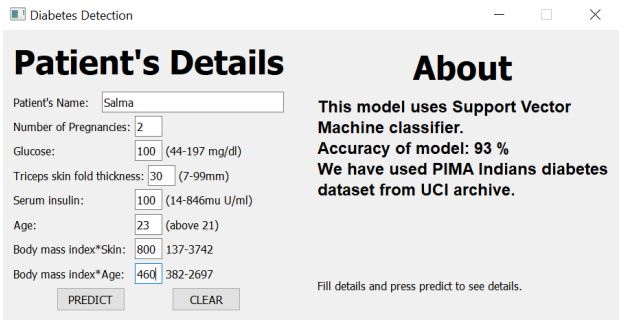

Figure 9. Diabetes Detection Application With Input Data.

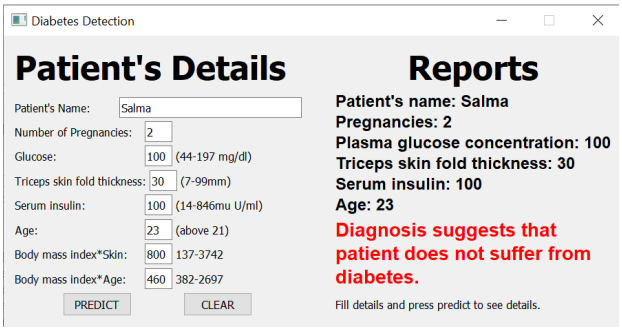

Figure 10. Diabetes Detection Application With Output of Prediction.

The user inputs the relevant information inside each box and the data is validated to be within the range. The user

clicks the PREDICT button to output the diabetes prediction diagnosis. The CLEAR button removes all the existing data entries to enter new input. The about explains further what this model uses and which dataset it was trained on. We changed the color of the diagnosis prediction to emphasize it in which the red color represents diabetic prediction and green represents non-diabatic prediction and we adjusted the font to make it bigger than the rest of the information in the outputted report.

4. Preprocessing Components

Once the user finishes inserting the required features, the added observation is merged with the original data frame. Normalization of the numerical features using a MinMax scaler is performed for the new observation. Then, the number of points is assigned to each class (0: non-diabatic; 1: diabatic).

5. Data Validation

The data validation was handled and the corresponding error messages for each entry are as described below:

1. Pregnancies:

- Data validation ensures that the value entered is not less than zero.

- Error message: "Pregnancies must be greater than or equal to zero."

2. Glucose:

- Data validation ensures that the value entered falls within the range of 44 to 197 mg/dl.

- Error message: "Glucose must be greater than 40 and less than 197 mg/dl."

3. Triceps skin fold thickness :

- Data validation ensures that the value entered falls within the range of 7 to 99 mm.

- Error message: "Skin thickness must be greater than 7 and less than 99 mm."

4. Serum insulin:

- Data validation ensures that the value entered falls within the range of 14 to 846.

- Error message: "Serum insulin must be greater than 14 and less than 846."

5. Age:

- Data validation ensures that the value entered is not less than 21.

- Error message: "Age must be greater than 21."

6. BMIxThickness:

- Data validation ensures that the value entered falls within the range of 137 to 3742.

- Error message: "BMIxThickness must be greater than 137 and less than 3742."

7. BMIxAge:

- Data validation ensures that the value entered falls within the range of 382 to 2697.

- Error message: "BMIxAge must be greater than 382 and less than 2697."

If any of the entered values fail to meet the specified criteria, a message box with the corresponding error message is displayed to the user, indicating which field needs correction. This ensures that only valid input data is processed for diabetes prediction.

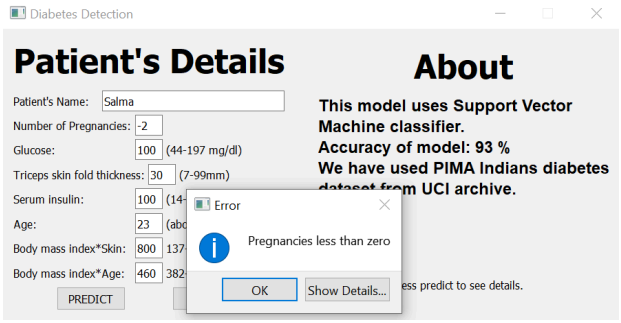

Figure 11. Diabetes Detection Application With Output of Prediction Showing Error Message.

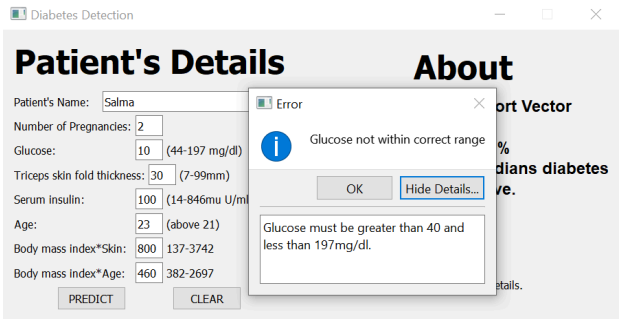

Figure 12. Diabetes Detection Application With Output of Prediction Showing Error Messages.

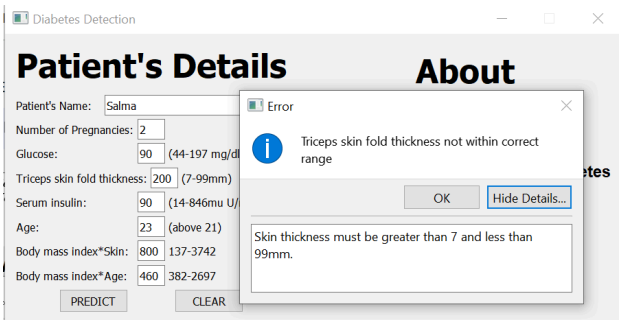

Figure 13. Diabetes Detection Application With Output of Prediction Showing Error Message.

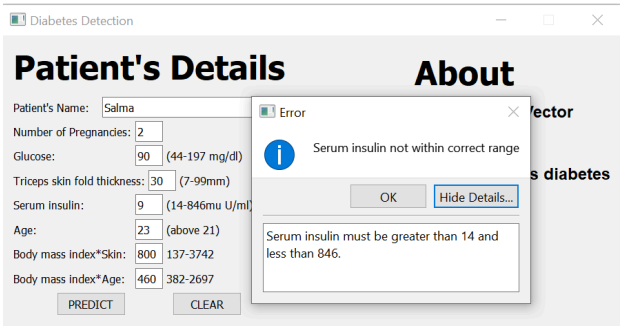

Figure 14. Diabetes Detection Application With Output of Prediction Showing Error Message.

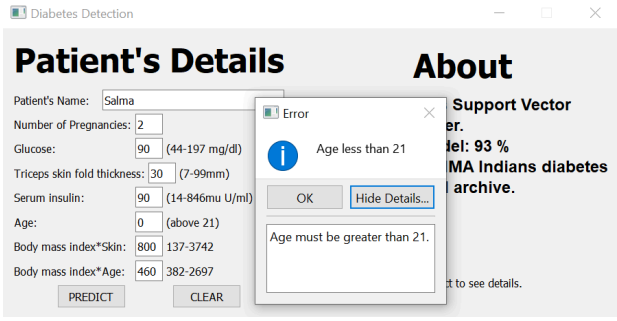

Figure 15. Diabetes Detection Application With Output of Prediction Showing Error Message.

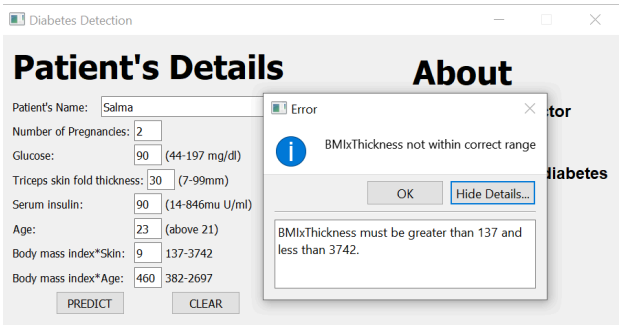

Figure 16. Diabetes Detection Application With Output of Prediction Showing Error Message.

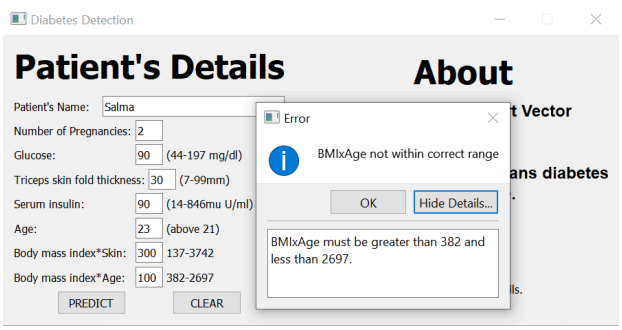

Figure 17. Diabetes Detection Application With Output of Prediction Showing Error Message.

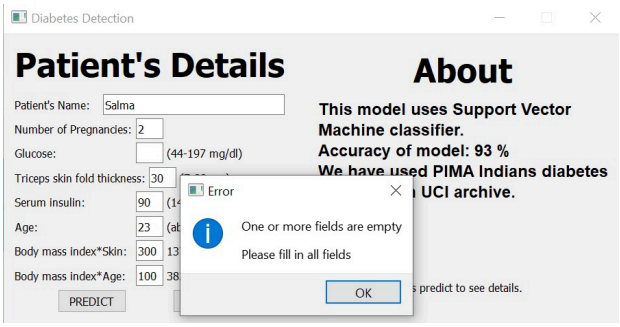

Figure 18. Diabetes Detection Application With Output of Prediction Showing Error Message.

If any of the fields are empty, a message box is displayed with an error icon. The message within the box states: “One or more fields are empty. Please fill in all fields.”This part ensures that the user cannot proceed with prediction until all required fields are filled in, enhancing the usability and reliability of the application.

REFERENCES

[1] Analytics Vidhya. (2017, September). Understanding Support Vector Machine(SVM) algorithm from examples (along with code). Retrieved from https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[2] Dua, D., & Taniskidou, E. K.,2018,Kaggle Machine Learning Repository, “Pima Indians Diabetes Dataset”, [Online]: Retrieved from: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database/data

[3] ODE-based Epidemic Network Simulation of Viral Hepatitis A and Kernel Support Vector Machine based Vaccination Effect Analysis - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Four-Kernel-Functions-for-Kernel-based-SVM_tbl3_341330102

[4] *Python Data Analysis Cookbook*. Retrieved March 27, 2024 from https://subscription.packtpub.com/book/data/9781785282287/10/ch10lvl1sec133/computing-precision-recall-and-f1-score