Diabetes Prediction Using Machine Learning

DSCI3415 Project Phase III

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

ABSTRACT

This paper addresses the global health crisis of diabetes, a chronic illness affecting millions worldwide. If accurate early prediction of the illness is achieved, the risk factor and severity of diabetes can be considerably decreased. This project aims to create a system capable of early diabetes prediction for patients, with increased accuracy through the integration of outcomes from diverse machine learning techniques. Employing algorithms such as K-nearest neighbour, Logistic Regression, Random Forest, Support Vector Machine, and Decision Tree, the model's accuracy is assessed for each algorithm. Subsequently, the algorithm demonstrating notable accuracy is selected as the predictive model for type 2 diabetes. Through applying different classifiers on the Pima Indians Diabetes Dataset, this paper explores the current landscape of employing machine learning techniques in predicting type 2 diabetes.

FINAL LIST OF CHOSEN FEATURES

	Pregnancies	Glucose	SkinThickness	Insulin	Age	BMIxThickness	BMIxAge	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	0.226180	0.501142	0.240107	0.153550	0.204015	0.233260	0.301627	0.348958
std	0.198210	0.196543	0.096639	0.107092	0.196004	0.125907	0.189209	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.058824	0.359677	0.195652	0.106370	0.050000	0.146290	0.156007	0.000000
50%	0.176471	0.470968	0.228261	0.106370	0.133333	0.218169	0.262290	0.000000
75%	0.352941	0.620968	0.271739	0.186899	0.333333	0.295569	0.421203	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Table 1. Description of the final clean dataset.

Our final clean dataset contains 8 main features. The original features Glucose, Insulin, SkinThickness, Age and Outcome, and the engineered features BMIxThickness and BMIxAge. We chose these as the final list of features as accordng to Extra Trees, K-Best Features, and Logistic Regression they had the highest relative importance/scores.

```
final_df.shape
```

```
(768, 8)
```

Figure 1. Size of the final clean dataset.

Our new dataset now has 768 observations, as we did not lose any of our observations. It has 8 features, including the target feature “Outcome”. It does not have any missing values nor incorrect zero values. Our final dataset is also normalized in order to produce the most accurate results.

DESCRIPTIVE STATISTICS

A summary of the descriptive statistics is provided in the tables below for each group (diabetic or non-diabetic).

	Pregnancies	Glucose	SkinThickness	Insulin	Age	BMIxThickness	BMIxAge	Outcome
count	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	268.0
mean	0.286216	0.634208	0.279040	0.208673	0.267786	0.289330	0.395828	1.0
std	0.220073	0.190250	0.092056	0.113118	0.182804	0.127671	0.175484	0.0
min	0.000000	0.219355	0.000000	0.000000	0.000000	0.015534	0.062424	1.0
25%	0.102941	0.483871	0.250000	0.186899	0.116667	0.221123	0.253013	1.0
50%	0.235294	0.619355	0.271739	0.186899	0.250000	0.263759	0.386038	1.0
75%	0.470588	0.793548	0.315217	0.186899	0.383333	0.342746	0.501847	1.0
max	1.000000	1.000000	1.000000	1.000000	0.816667	1.000000	0.949240	1.0

Figure 2: Summary statistic of numerical variables for group 1 of target = 1

	Pregnancies	Glucose	SkinThickness	Insulin	Age	BMIxThickness	BMIxAge	Outcome
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.0
mean	0.194000	0.429819	0.219239	0.124005	0.169833	0.203207	0.251135	0.0
std	0.177481	0.159380	0.092575	0.090968	0.194461	0.114243	0.176720	0.0
min	0.000000	0.000000	0.000000	0.001202	0.000000	0.000000	0.000000	0.0
25%	0.058824	0.316129	0.163043	0.097356	0.033333	0.125964	0.115582	0.0
50%	0.117647	0.406452	0.217391	0.106370	0.100000	0.185132	0.206584	0.0
75%	0.294118	0.522581	0.260870	0.109375	0.266667	0.255589	0.334521	0.0
max	0.764706	0.987097	0.576087	0.877404	1.000000	0.740860	1.000000	0.0

Figure 3: Summary statistic of numerical variables for group 2 of target = 0

EXPERIMENTAL ANALYSIS

Splitting was performed to include 80% of the observations for training and the other 20% to test the model performance using the function *train_test_split* from *sklearn* package in Python. Shuffling was set to *True*. We also applied hyperparameter tuning and model selection using cross-validation with *sklearn* package. 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.

Nine models were trained using the default parameters using *sklearn*: KNN, Support Vector Machine, Logistic Regression, Decision Tree, Gradient Boosting, XGB Classifier, AdaBoostClassifier, Random Forest, and Naive Bayes. 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.

Models' Performance Evaluation: Precision, recall, accuracy and F1 score were calculated for each model using all observations for each group (diabetic or not diabetic).

1) Logistic Regression

Logistic regression is used to describe data and the relationship between one dependent variable and one or more independent variables.

Parameters Used: L2 regularization and C = 1. A higher C means less regularization strength which can lead to overfitting easily. A lower C means higher regularization strength which means that the fitted line is robust to outliers and will not change much by the highly influential observations. Default parameters were chosen to run the classifier.

Classification Report is:

	precision	recall	f1-score	support
0	0.72	0.90	0.80	89
1	0.79	0.52	0.63	65
accuracy			0.74	154
macro avg	0.76	0.71	0.71	154
weighted avg	0.75	0.74	0.73	154

Figure 4. Classification report of logistic regression.

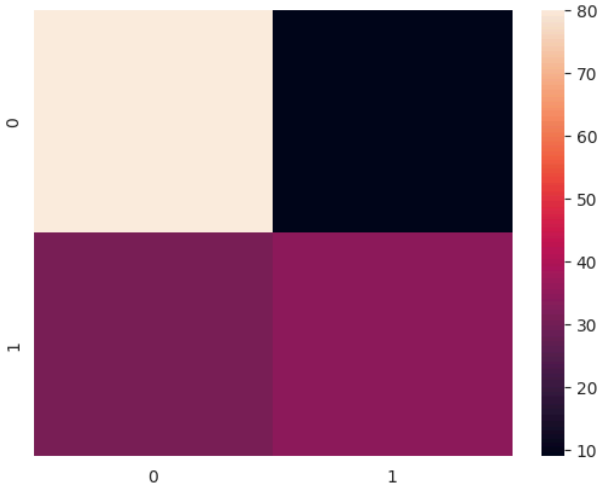

Figure 5. Confusion matrix of logistic regression.

The precision, recall, and accuracy scores are relatively low, and therefore it shows that the logistic regression performed poorly with the dataset. Logistic regression assumes a linear relationship between features and the target variable, and so maybe the relationship in the data is non-linear, so the logistic regression did not perform well.

Pros:

1. Simple and interpretable: Logistic regression is a simple linear model that provides interpretable results. It's easy to understand how each feature contributes to the prediction.
2. Low risk of overfitting: Logistic regression tends to generalize well and has a low risk of overfitting, especially with small datasets, as it's less prone to capturing noise.
3. Well-suited for binary classification

Cons:

1. Assumption of linearity: Logistic regression assumes that the relationship between the independent variables and the dependent variable is linear. If this assumption is violated, the model may not perform well.
2. Limited expressiveness: Logistic regression is a linear model, so it may not capture complex relationships between features as effectively as more complex algorithms like decision trees or neural networks.
3. Cannot handle non-linear relationships: Logistic regression cannot capture non-linear relationships between features and the target variable, and our dataset might have non-linear relationships.

2) Decision Tree

Parameters used: criterion: gini (a powerful measure of the randomness or the impurity or entropy in the values of a dataset). The maximum depth of the tree was not specified, which means that the tree nodes are expanded until the leaves are pure. Also, the minimum number of samples required to split an internal node was set to 2. However, the decision tree model sometimes makes a split, including one observation in a branch. Other times, it is susceptible to noise, and overfitting could happen.

In order to optimize the hyperparameters for the decision tree, Grid Search CV algorithm was used to find out the optimal hyperparameters and use them. After using the optimized hyperparameters and rerunning the classifier, the accuracy scores improved.

Classification Report is:

	precision	recall	f1-score	support
0	0.88	0.93	0.91	89
1	0.90	0.83	0.86	65
accuracy			0.89	154
macro avg	0.89	0.88	0.89	154
weighted avg	0.89	0.89	0.89	154

Figure 6. Classification report of the decision tree.

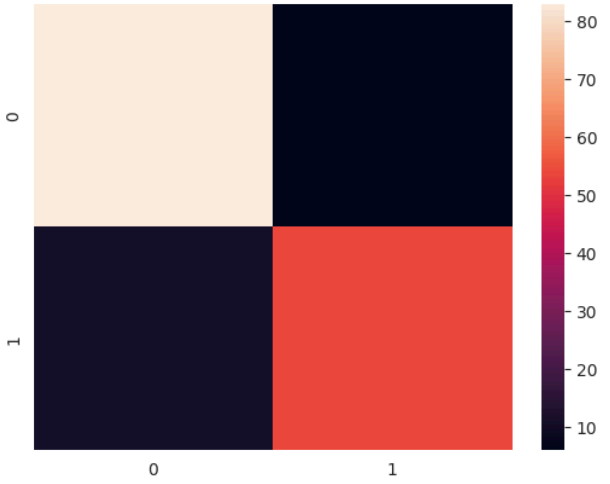

Figure 7. Confusion matrix of decision tree.

The decision tree's precision, recall and accuracy scores are relatively high which shows that the classifier performed well with the dataset. Decision trees are capable of capturing non-linear relationships between features and the target variable inherently, the dataset might have non-linear relationships so this could be an explanation for why decision trees might perform better.

Pros:

1. Non-linearity: Decision trees are capable of capturing non-linear relationships between features and the target variable, which may exist in the Pima Indian dataset.
2. Interpretability: Decision trees offer a transparent and intuitive representation of decision-making. They are easy to visualize and understand, making them useful for explaining the reasoning behind predictions.
3. Automatic feature selection: Decision trees inherently perform feature selection by selecting the most informative features at each split, which can be advantageous for our datasets that might have irrelevant features.

Cons:

1. Overfitting: Decision trees are prone to overfitting, especially when the tree depth is not properly controlled. In the case of the Pima Indian dataset, overfitting may occur if the tree becomes too complex and captures noise in the data.
2. Instability: Decision trees can be sensitive to small variations in the data, leading to different trees being generated on different runs or subsets of the data. This instability can affect the reliability of the model.
3. Limited generalization: While decision trees can capture complex relationships within the training data, they may not generalize well to unseen data, especially if the relationships are highly specific to the training data and do not hold in the broader population. Our dataset it a bit small, so the classifier might memorize the data points.
4. Difficulty with continuous variables: Decision trees may not perform well with continuous variables, especially if the relationships are not easily split by simple thresholding. This could potentially limit their effectiveness with certain features in the Pima Indian dataset.

3) Random Forest

Parameters used: number of estimators was set to 100. The Gini criterion and other parameters for the forest were set as in the previous model (decision tree).

Grid Search CV was also used to optimize the hyperparameters.

Classification Report is:	precision	recall	f1-score	support
0	0.89	0.96	0.92	89
1	0.93	0.85	0.89	65
accuracy			0.91	154
macro avg	0.91	0.90	0.91	154
weighted avg	0.91	0.91	0.91	154

Figure 8. Classification report of random forest.

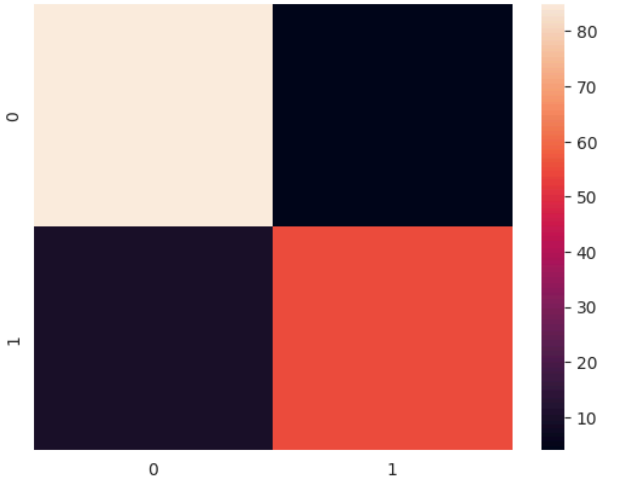

Figure 9. Confusion matrix of random forest.

Random forest's precision, recall, and accuracy scores are relatively high which shows that the classifier performed well with the dataset. Similar to the decision tree, it is possible to have a good performance due to the relationship between features in the dataset.

Pros:

1. High predictive accuracy: Random Forest tends to provide high predictive accuracy compared to individual decision trees, as it aggregates the predictions of multiple trees. This can be advantageous for datasets like Pima Indian where accurate predictions are crucial.
2. Non-linearity and interactions: Random Forest can capture non-linear relationships and interactions between features effectively, which may exist in the Pima Indian dataset but may not be captured well by linear models.
3. Robustness to overfitting: Random Forest is less prone to overfitting compared to individual decision trees, especially when the number of trees in the forest is properly tuned. This can help prevent the model from capturing noise in the data and improve generalization performance.

Cons:

1. Complexity and interpretability: Random Forest models are more complex than individual decision trees, making them less interpretable. Understanding the decision-making process behind Random Forest predictions may be more challenging compared to decision trees.
2. Not suitable for real-time prediction: Random Forest models are not well-suited for real-time prediction tasks due to their computational complexity and the need to aggregate predictions from multiple trees.

4) Naive Bayes

Used parameters: Portion of the largest variance of all features that is added to variances for calculation stability was set to 1e-9 as the default parameter.

Classification Report is:	precision	recall	f1-score	support
0	0.73	0.87	0.79	89
1	0.76	0.57	0.65	65
accuracy			0.74	154
macro avg	0.74	0.72	0.72	154
weighted avg	0.74	0.74	0.73	154

Figure 10. Classification report of naive bayes.

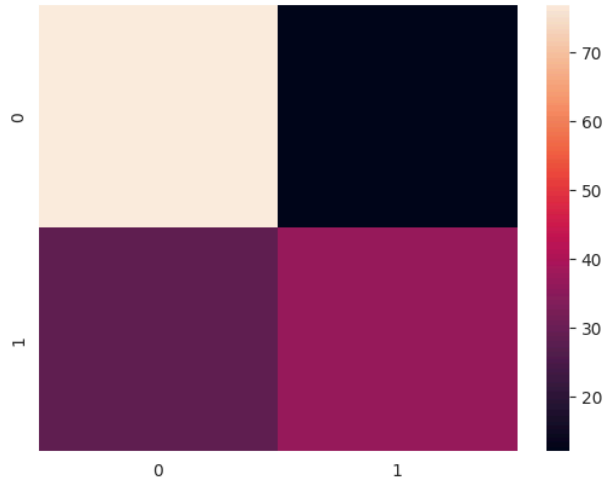

Figure 11. Confusion matrix of naive bayes.

Naive Bayes produced relatively low precision, recall, and accuracy scores.

Pros:

1. Robust to noise: Naive Bayes is robust to noisy data and can perform well even when the independence assumption is not entirely met. It can still provide reasonable predictions in the presence of irrelevant or correlated features.
2. Effective with small datasets: Naive Bayes tends

to perform well with small datasets, such as the Pima Indian dataset, where it can provide reliable predictions without requiring extensive computational resources or data preprocessing.

Cons:

1. Strong independence assumption: Naive Bayes assumes that features are independent given the class label, which may not hold true in real-world datasets like Pima Indian. Violation of this assumption can lead to suboptimal performance.
2. Limited expressiveness: Due to its simplistic nature, Naive Bayes may not capture complex relationships between features as effectively as more sophisticated algorithms like Random Forest or Gradient Boosting Machines.
3. Cannot handle interactions between features: Naive Bayes does not model interactions between features, which may be important in datasets like Pima Indian where interactions between certain risk factors for diabetes may exist.

5) SVM

Parameters used: the regularization parameter C was set to 1 (as the default value), the same as in the logistic regression mentioned above. The rbf kernel was used. The degree was set to 3. Then we calculated the accuracy score, and it turned out to be 0.79.

We wanted to improve the score, so Grid Search CV algorithm was used and the SVM classifier was repeated to observe the new accuracy score. The new accuracy score is 0.94, which shows that the Grid Search algorithm significantly affected the accuracy score.

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 12. Classification report of SVM.

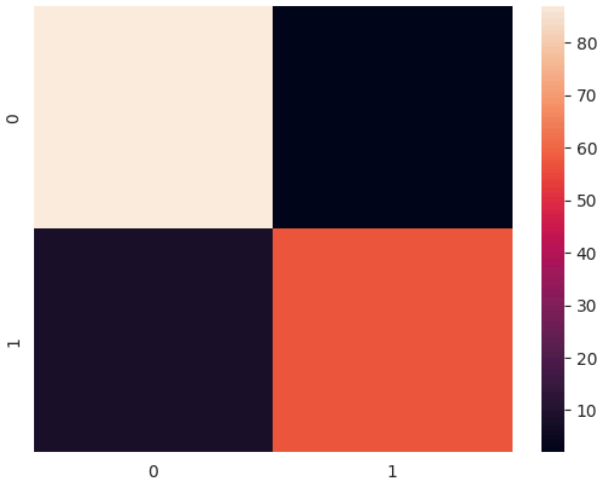

Figure 13. Confusion matrix of SVM.

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.

6) AdaBoostClassifier (100 estimators)

Parameters used: The maximum number of estimators at which boosting is ended was set to 50. The learning rate was set to 1.

Classification Report is:

	precision	recall	f1-score	support
0	0.87	0.96	0.91	89
1	0.93	0.80	0.86	65
accuracy			0.89	154
macro avg	0.90	0.88	0.88	154
weighted avg	0.89	0.89	0.89	154

Figure 14. Classification report of AdaBoostClassifier.

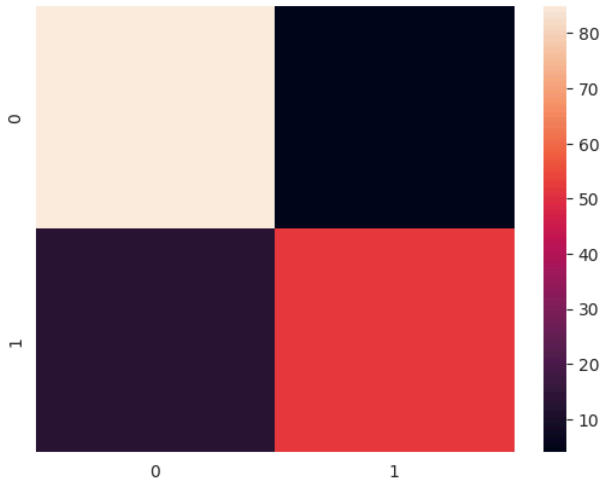

Figure 15. Confusion matrix of AdaBoostClassifier.

Pros:

1. Improved predictive performance: AdaBoost often yields better predictive performance compared to individual weak learners, as it focuses more on instances that are difficult to classify correctly. This can be beneficial for datasets like Pima Indian where accurate predictions are crucial.
2. Automatic feature selection: AdaBoost can implicitly perform feature selection by giving more weight to informative features that contribute to the classification task, leading to simpler and more interpretable models.
3. Robustness to overfitting: AdaBoost is less prone to overfitting compared to individual weak learners, especially when using simple base learners like decision trees. It tends to generalize well to unseen data.

Cons:

1. Sensitive to noisy data and outliers: AdaBoost can be sensitive to noisy data and outliers, as it may attempt to fit the noisy instances during the boosting process, leading to suboptimal performance.
2. Bias towards more complex models: AdaBoost tends to favor more complex models as base learners, which may lead to increased model complexity and decreased interpretability, especially when using decision trees with large depths.
3. Limited interpretability: The final AdaBoost model may be less interpretable compared to individual base learners, as it combines multiple weak learners into a single ensemble model, making it harder to understand the underlying decision-making process.

7) KNN

Parameters used: A grid search was performed to identify the best value of k that would give a higher F1 score, as shown below. Observations in the neighborhood were not weighted. Euclidean distance was used to compare the distances between each pair of observations. However, KNN is memory intensive as it needs to store all observations in memory to predict the class of the new observation after calculating distances between the new one and each of the training set.

After Grid Search CV, the best leaf size turned out to be 30, the best p is 1, and the best n_neighbors (k) is 19.

Classification Report is:

	precision	recall	f1-score	support
0	0.82	0.90	0.86	89
1	0.84	0.74	0.79	65
accuracy			0.83	154
macro avg	0.83	0.82	0.82	154
weighted avg	0.83	0.83	0.83	154

Figure 16. Classification report of KNN

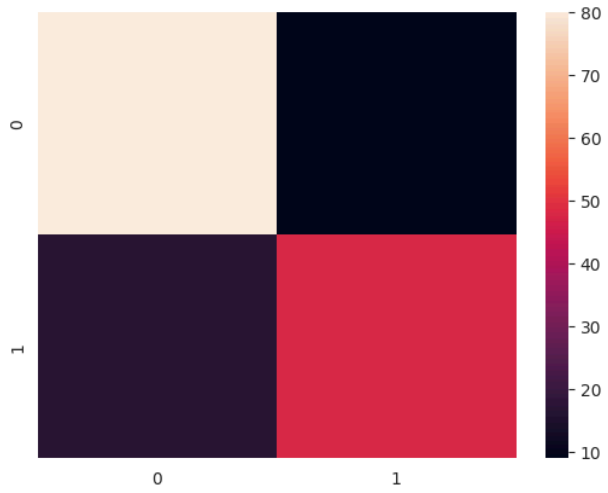

Figure 17. Confusion matrix of KNN (k =19).

KNN yielded relatively high precision, recall and accuracy scores.

Pros:

1. Simple to understand and implement: KNN is an intuitive algorithm that is easy to understand and implement, making it useful for quick prototyping.
2. No training phase: KNN is an instance-based learning algorithm, which means it doesn't require an explicit training phase. Instead, it stores all training data points and performs computation at the time of prediction. Which could help us when deploying the model in a hospital for example. The patient would enter their symptoms and immediately receive a prediction.

Cons:

1. Sensitive to irrelevant features and scaling: KNN is sensitive to irrelevant features and features with different scales. Irrelevant features can introduce noise, while features with different scales can lead to biased distance calculations. Proper feature selection and scaling are necessary for optimal performance.
2. Memory-intensive: As KNN stores all training data points, it can be memory-intensive, especially for large datasets. Memory constraints may limit its applicability to datasets that cannot fit into memory.
3. Limited interpretability: KNN provides little insight into the underlying decision-making process. It doesn't offer explicit explanations or feature importance rankings, which can make it challenging to interpret the model's predictions.

KNN is a simple and versatile algorithm suitable for the Pima Indian Diabetes dataset. However, its computational inefficiency, sensitivity to irrelevant features, and lack of interpretability should be considered when trying to deploy the model for diabetes prediction.

8) Gradient Boosting

The default parameters were used for gradient boosting.

Classification Report is:

	precision	recall	f1-score	support
0	0.86	0.97	0.91	89
1	0.94	0.78	0.86	65
accuracy			0.89	154
macro avg	0.90	0.88	0.88	154
weighted avg	0.90	0.89	0.89	154

Figure 18. Classification report of Gradient Boosting.

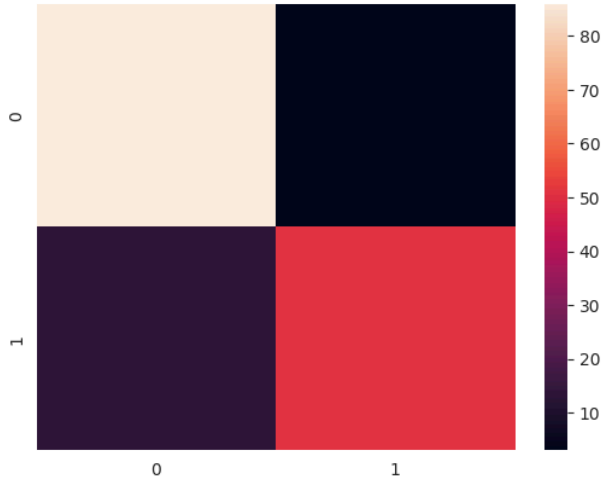

Figure 19. Confusion Matrix of Gradient Boosting.

Gradient Boost produced relatively high precision, recall, and accuracy scores.

Pros:

1. High predictive accuracy: Gradient boosting typically yields high predictive accuracy by combining the predictions of multiple weak learners. It sequentially improves the model's performance by focusing on the errors made by the previous models, leading to better overall predictions.
2. Handles complex relationships: Gradient boosting can capture complex relationships between features and the target variable, making it suitable for datasets like Pima Indian where the relationships may be nonlinear or intricate.
3. Robustness to overfitting: Gradient boosting

tends to be less prone to overfitting compared to other ensemble methods like bagging due to its sequential nature. It generalizes well to unseen data by continuously refining the model.

4. Feature importance: Gradient boosting provides insights into feature importance, helping to identify which features are most influential in making predictions. This can aid in feature selection and interpretation.

Cons:

1. Sensitivity to noisy data and outliers: Gradient boosting can be sensitive to noisy data and outliers, as it may attempt to fit the errors made by previous models. Outliers and noisy data points can disproportionately influence subsequent iterations, leading to suboptimal performance.
2. Potential for overfitting with deep trees: While gradient boosting is less prone to overfitting compared to some other algorithms, using very deep trees in the ensemble can still lead to overfitting, especially on smaller datasets like Pima Indian.
3. Less interpretable compared to simpler models: Gradient boosting models are less interpretable compared to simpler models like logistic regression or decision trees. Understanding the precise reasoning behind individual predictions can be challenging due to the ensemble nature of the model.

9) XGBoost Classifier

The default parameters were used for XGBoost classifier.

Classification Report is:

	precision	recall	f1-score	support
0	0.84	0.97	0.90	89
1	0.94	0.75	0.84	65
accuracy			0.88	154
macro avg	0.89	0.86	0.87	154
weighted avg	0.88	0.88	0.87	154

Figure 20. Classification report of XGB Classifier.

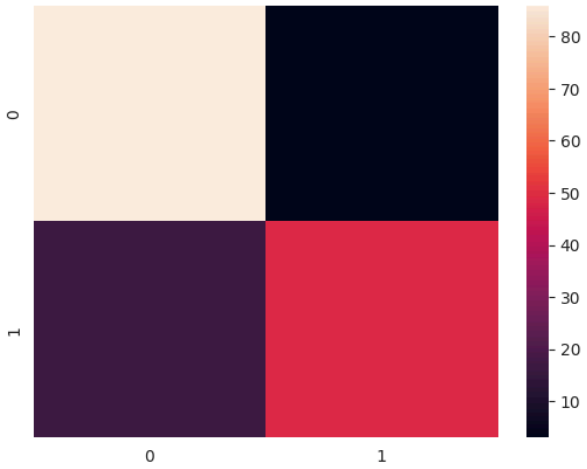

Figure 21. Confusion matrix of XGB Classifier.

Pros:

1. High predictive accuracy: XGBoost often achieves state-of-the-art performance on various datasets, including the Pima Indian Diabetes dataset. It leverages gradient boosting techniques to sequentially improve model predictions, leading to high accuracy.
2. Robustness to overfitting: XGBoost includes regularization techniques such as shrinkage (learning rate) and feature subsampling, which help prevent overfitting and improve the model's generalization ability.
3. Feature importance: XGBoost provides insights into feature importance, allowing users to understand which features contribute the most to the model's predictions. This can aid in feature selection and interpretation.

Cons:

1. Complexity and tuning: XGBoost has several hyperparameters that need to be tuned, such as the learning rate, tree depth, and regularization parameters. Finding the optimal combination of hyperparameters can be time-consuming and require extensive experimentation.
2. Potential for overfitting with deep trees: While XGBoost includes regularization techniques to mitigate overfitting, using very deep trees in the ensemble can still lead to overfitting, especially on smaller datasets like Pima Indian.
3. Less interpretable compared to simpler models: XGBoost models are less interpretable compared to simpler models like logistic regression or decision trees. Understanding the precise reasoning behind individual predictions can be challenging due to the ensemble nature of the model.

COMPARATIVE ANALYSIS OF TOP 3 MODELS

After implementing the nine classifiers, we compared their performance to determine the best 3 among them that fit well with the Pima Indians Dataset.

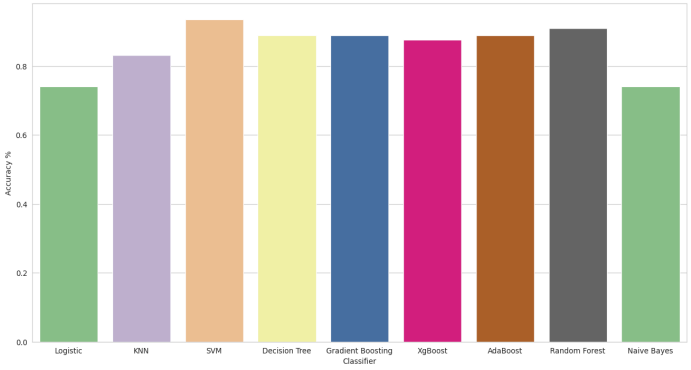

Figure 22. Chart for Accuracy score of all 9 classifiers.

Model	Accuracy	Precision	Recall	F1
Logistic regression	**0.74**	**0.76**	**0.71**	**0.71**
Decision Tree	**0.89**	**0.90**	**0.83**	**0.86**
Random Forest	**0.91**	**0.93**	**0.85**	**0.89**
Naive Bayes	**0.74**	**0.74**	**0.72**	**0.72**
SVM	**0.94**	**0.97**	**0.88**	**0.92**
AdaBoostClassifier (100 estimators)	**0.89**	**0.90**	**0.88**	**0.88**
KNN (k = 19)	**0.83**	**0.83**	**0.82**	**0.82**
Gradient Boosting	**0.89**	**0.90**	**0.88**	**0.88**
XgBoost	**0.88**	**0.89**	**0.86**	**0.87**

As shown from the graphs and table above, we identified the top 3 models which are SVM, Random Forest, and Decision Tree respectively using Accuracy, Precision, Recall, and F1 score to asses the models' performance.

Top 3 models

1. Support Vector Machine
2. Random Forest
3. Decision Tree

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + T_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figure 23. Accuracy, Precision, Recall,and F1 Metric Formula

- **Accuracy**

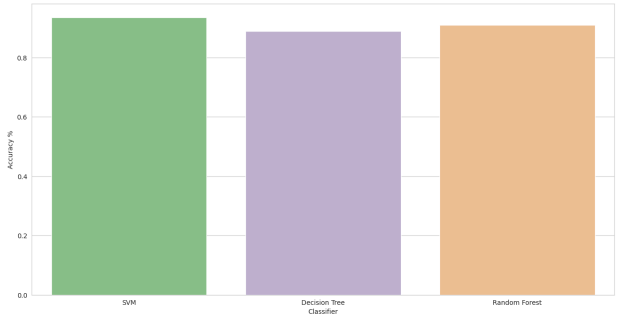

Figure 24. Accuracy of top 3 classifiers.

The accuracy metric is the ratio of correct predictions. In our analysis, SVM showed the highest accuracy (0.94), followed by Random Forest (0.91) and Decision Tree (0.89). The variability in accuracy might be due to the different decision boundaries each model constructs. SVM tends to find the hyperplane that best separates the classes, while Random Forest and Decision Tree create decision trees based on feature splits, which may lead to overfitting.

- **Precision**

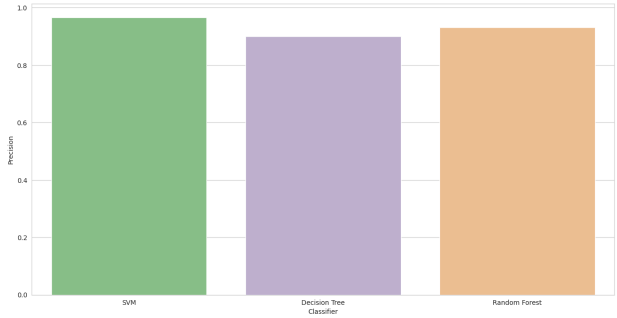

Figure 25. Precision of top 3 classifiers.

The precision metric is the ratio of true positives to the

total positive predictions made by the model. It measures the model's ability to avoid false positives which in our case is very significant as we do not want to classify a patient as non-diabetic while this patient is originally diabetic. In our case, SVM exhibited the highest precision (0.97), followed by Random Forest (0.93) and Decision Tree (0.90). The variability in precision could be due to SVM's ability to construct a robust margin that maximizes the separation between classes, thereby reducing false positives.

- **Recall**

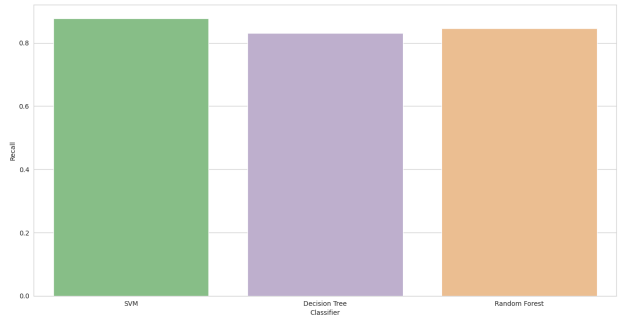

Figure 26. Recall of top 3 classifiers.

The recall metric measures the ratio of true positive predictions to the actual positive instances in the dataset. It shows the model's ability to capture all positive instances. SVM achieved the highest recall (0.88), followed closely by Random Forest (0.85) and Decision Tree (0.83). The variability in recall might stem from the different strategies each model employs in constructing decision boundaries and handling class imbalance.

The F1 score computes the average of precision, and recall computes the average of precision and recall, providing a balance between the two metrics. SVM demonstrated the highest F1 score (0.92), followed by Random Forest (0.89) and Decision Tree (0.86). The variability in the F1 score could be due to the trade-off between precision and recall in each model's decision-making process.

CHOSEN TECHNIQUE

SVM will be chosen as it provides the highest accuracy, precision, and recall scores. It 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. SVM has the ability to find the optimal hyperplane for separating classes, even in non-linearly separable cases, through kernel trick. 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). 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.

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 27. Classification report of SVM.

REFERENCES

[1] 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

[2[2.*Python Data Analysis Cookbook*. Retrieved March 27, 2024 from https://subscription.packtpub.com/book/data/9781785282287/10/ch10lvl1sec133/computing-precision-recall-and-f1-score