Diabetes Prediction using Machine Learning in Python

A MINOR-I PROJECT REPORT

Submitted by

ADEEB UR REHMAN 2020-310-011

in partial fulfilment for the award of the degree of

B. TECH COMPUTER SCIENCE & ENGINEERING

Under the supervision of

Dr. Sameena Naaz

Department of Computer Science & Engineering
School of Engineering Sciences & Technology
JAMIA HAMDARD
New Delhi-110062

(2023)

DECLARATION

I, **Mr. Adeeb Ur Rehman** a student of **Bachelors of Technology Computer Science & Engineering (B. Tech CSE), Enrolment No: 2020-310-011)** hereby declare that the Project/Dissertation entitled "**Diabetes Prediction using Machine Learning in Python"** which is being submitted by me to the Department of Computer Science, Jamia Hamdard, New Delhi in partial fulfilment of the requirement for the award of the degree of **Bachelors of Technology Computer Science & Engineering (B. Tech CSE),** is my original work and has not been submitted anywhere else for the award of any Degree, Diploma, Associateship, Fellowship or other similar title or recognition.

ADEEB UR REHMAN

2020-310-011

Date: April 2023

Place: New Delhi

ACKNOWLEDGEMENT

I express my sincere thanks to **Dr Sameena Naaz** (Dept. of Computer Science and Engineering), my project in charge, who guided me through the project for her valuable suggestions and guidance for completing the project. This project has been a success only because of h er guidance.

I deeply express my sincere thanks to our Head of Department **Dr. Farheen Siddiqui** for encouraging me and allowing me to present the project on the topic "**Diabetes Prediction using Machine Learning in Python**" at the department premises for the partial fulfilment of the requirements leading to the award of B.Tech degree.

I am also thankful to the whole computer science and engineering department for providing the technical support to carry out the project work, letting us utilize all the necessary facilities of the institute and providing guidance at each& every step during the project work.

Adeeb Ur Rehman

2020-310-011

INDEX

OBJECTIVE

The objective of this machine learning project is to develop a predictive model that can accurately predict the likelihood of diabetes in patients using a diabetes dataset. The model should be able to analyse various features such as age, BMI, blood pressure, insulin levels, and others to accurately classify patients as either diabetic or non-diabetic.

The project aims to leverage machine learning algorithms to improve early detection of diabetes, which could lead to better patient outcomes through early intervention and management of the disease.

The project may also involve feature selection, model evaluation, and optimization to achieve the highest possible accuracy and interpretability in the predictions. The goal is to create a reliable and scalable model that can be used in clinical settings to aid healthcare providers in making informed decisions about diabetes diagnosis and treatment.

INTRODUCTION

Diabetes, a chronic metabolic disorder, has become a significant health concern worldwide. It affects millions of people, and its complications can lead to severe health issues, including cardiovascular diseases, neuropathy, retinopathy, and kidney failure. Early and accurate prediction of diabetes can play a crucial role in its management and prevention of complications. Machine learning algorithms, such as Support Vector Machine (SVM) and Random Forest Classification, have emerged as promising tools for diabetes prediction due to their ability to handle complex datasets and make accurate predictions.

Machine learning is a field of artificial intelligence (AI) that involves the development of algorithms and models that allow computers to learn from and make predictions or decisions based on data, without being explicitly programmed. Machine learning algorithms are designed to identify patterns, relationships, and insights from large amounts of data, and they are widely used in various applications such as image recognition, natural language processing, recommendation systems, fraud detection, and more.

There are several types of machine learning algorithms, including:

1. **Supervised Learning**: In supervised learning, the algorithm is trained on labeled data, where the correct output or outcome is provided. The algorithm learns to make predictions or classifications based on this labeled data. Examples of supervised learning algorithms include linear regression, logistic regression, decision trees, and support vector machines (SVM).

2. **Unsupervised Learning**: In unsupervised learning, the algorithm is trained on unlabelled data, where the output or outcome is not provided. The algorithm identifies patterns, clusters, or relationships within the data without any labeled guidance. Examples of unsupervised learning algorithms include clustering.

3. **Reinforcement Learning**: In reinforcement learning, the algorithm learns through trial and error by taking actions in an environment and receiving feedback in the form of rewards or penalties based on its actions. The algorithm learns to make decisions or take actions that maximize the total cumulative reward over time. Reinforcement learning is widely used in robotics, game playing, and autonomous systems.

4. **Deep Learning**: Deep learning is a subset of machine learning that involves the use of artificial neural networks with multiple layers to process data and extract features automatically. Deep learning algorithms, such as convolutional neural networks (CNNs) for image recognition and recurrent neural networks (RNNs) for sequence data, have achieved state-of-the-art performance in many applications.

5. **Other Algorithms**: There are many other machine learning algorithms, such as decision trees, random forests, k-nearest neighbors (KNN), support vector machines (SVM), naive Bayes, and more, that are used for various tasks based on their strengths and weaknesses.

These are just some examples of the wide range of machine learning algorithms that are used in the field. The selection of the appropriate algorithm depends on the specific problem, dataset, and requirements of the task at hand. Machine learning algorithms continue to evolve, and researchers and practitioners are constantly developing new techniques and approaches to improve the performance and capabilities of machine learning models.

In this project, we aim to develop a diabetes prediction model using SVM and Random Forest Classification algorithms. We will leverage a dataset that contains various demographic, clinical, and lifestyle features of individuals, along with their diabetes status. The dataset will be preprocessed to handle missing values, feature scaling, and categorical variable encoding. We will then explore the data through visualisation and statistical analysis to gain insights into the characteristics of the dataset.

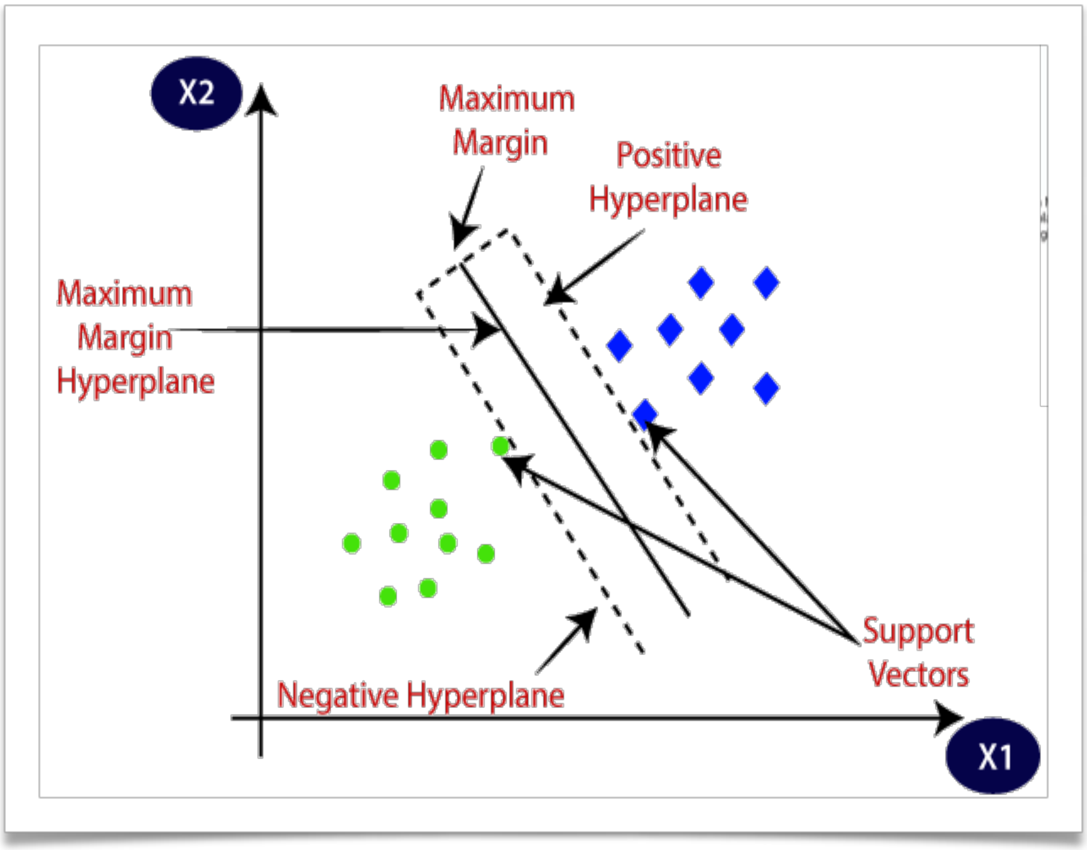
Next, we will implement the SVM and Random Forest Classification algorithms for diabetes prediction. SVM is a supervised learning algorithm that can classify data into two or more classes by finding an optimal hyperplane that maximally separates the classes. Random Forest Classification, on the other hand, is an ensemble learning algorithm that combines multiple decision trees to make predictions. Both SVM and Random Forest Classification have been proven to be effective in handling classification problems with high accuracy and have been widely used in various medical applications, including diabetes prediction.

Support Vector Machine Algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Applying Support Vector Machine (SVM) on a diabetes dataset involves several steps, including data preprocessing, model training, and model evaluation. Here's an overview of the process:

- **Data Preprocessing**: First, you need to load and preprocess the diabetes dataset. This typically involves importing the dataset into a suitable data structure (e.g., a DataFrame in Python), handling missing values, and splitting the dataset into features (X) and labels (y).

- **Feature Scaling**: SVM is sensitive to the scale of the input features, so it's important to scale them before training the model. You can use techniques like min-max scaling or standardization to scale the features to a similar range.

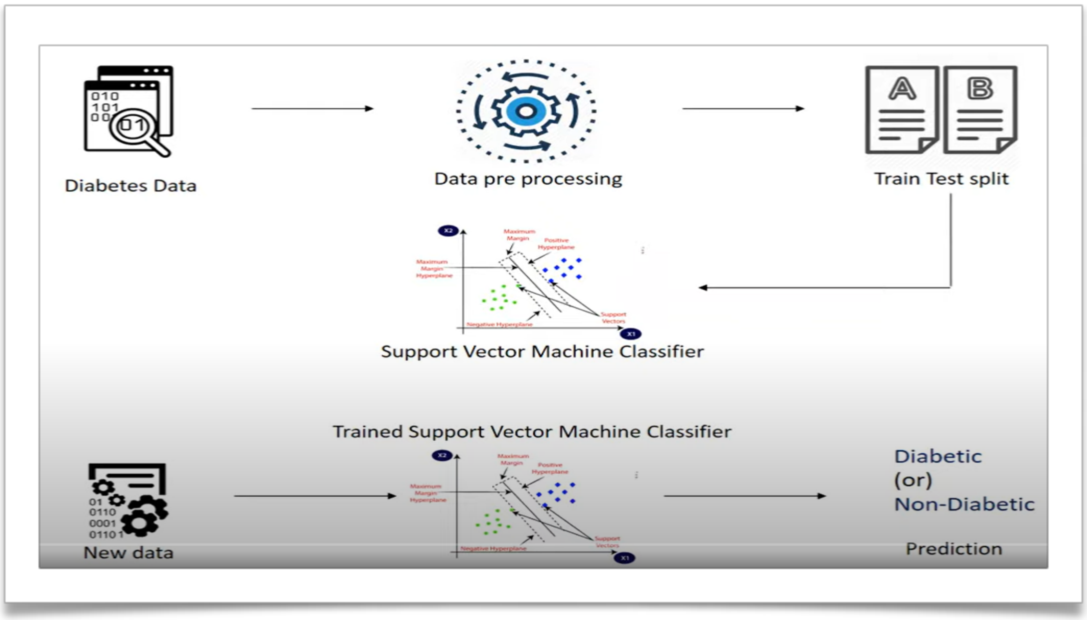
- **Splitting the Dataset**: Next, you need to split the dataset into training and testing sets. The training set is used to train the SVM model, while the testing set is used to evaluate the model's performance.

- **Model Training**: Once the dataset is preprocessed and split, you can train an SVM model on the training set. You can use the SVC class from the Scikit-learn library in Python to create an SVM classifier. You can specify the hyper-parameters of the SVM model, such as the kernel type (e.g., linear, polynomial, or radial basis function), regularisation parameter (C), and kernel-specific parameters (e.g., degree for polynomial kernel).

- **Model Evaluation**: After training the SVM model, you need to evaluate its performance on the testing set. You can use various evaluation metrics such as accuracy, precision, recall, and F1-score to assess the model's performance. You can also use techniques like cross-validation to get a more reliable estimate of the model's performance.

- **Hyper-parameter Tuning**: If the SVM model's performance is not satisfactory, you can perform hyper-parameter tuning to find the optimal hyper-parameter values that yield better results. Grid search or randomised search can be used for hyper-parameter tuning.

- **Model Deployment**: Once you are satisfied with the SVM model's performance, you can deploy the model to a production environment and use it for making predictions on new, unseen data.



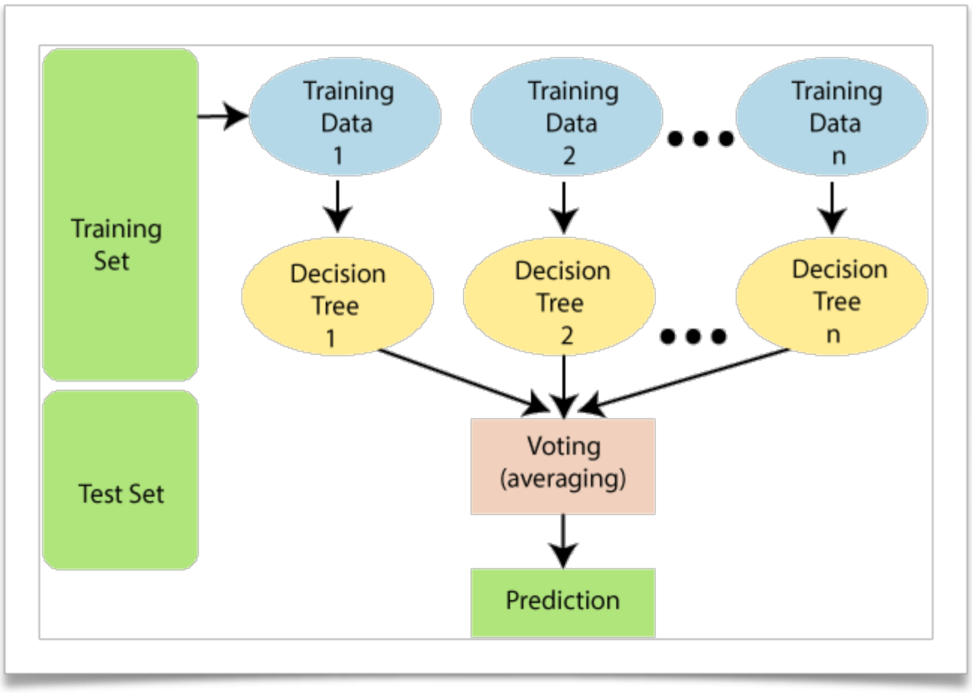
Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



Random Forest Classifier is an ensemble machine learning algorithm that can be used for classification tasks, including diabetes detection. Here's an overview of how you can apply the Random Forest Classifier on a diabetes dataset:

1. **Data Preprocessing**: First, you need to load and preprocess the diabetes dataset. This typically involves importing the dataset into a suitable data structure (e.g., a DataFrame in Python), handling missing values, and splitting the dataset into features (X) and labels (y).

2. **Feature Scaling**: Random Forest Classifier is a tree-based algorithm that is not sensitive to the scale of the input features, so feature scaling is generally not required.

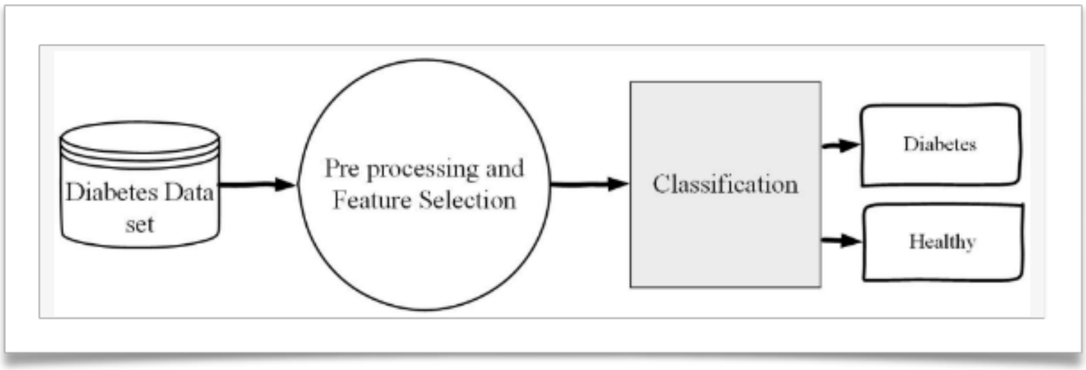
3. **Splitting the Dataset**: Next, you need to split the dataset into training and testing sets. The training set is used to train the Random Forest Classifier, while the testing set is used to evaluate the model's performance.

4. **Model Training**: Once the dataset is preprocessed and split, you can train a Random Forest Classifier on the training set. You can use the **RandomForestClassifier** class from the Scikit-learn library in Python to create a Random Forest Classifier. You can specify various hyper-parameters of the Random Forest Classifier, such as the number of trees (n_estimators), the maximum depth of trees (max_depth), the minimum number of samples required to split an internal node (min_samples_split), and others.

5. **Model Evaluation**: After training the Random Forest Classifier, you need to evaluate its performance on the testing set. You can use various evaluation metrics such as accuracy, precision, recall, and F1-score to assess the model's performance. You can also use techniques like cross-validation to get a more reliable estimate of the model's performance.

6. **Hyper-parameter Tuning**: If the Random Forest Classifier's performance is not satisfactory, you can perform hyper-parameter tuning to find the optimal hyper-parameter values that yield better results.

7. **Model Deployment**: Once you are satisfied with the Random Forest Classifier's performance, you can deploy the model to a production environment and use it for making predictions on new, unseen data.



Furthermore, we will explore feature selection techniques to identify the most relevant features that contribute to the prediction of diabetes. Feature selection is an essential step in machine learning as it helps to reduce the dimensionality of the dataset, improve model performance, and interpret the model's results.

Finally, we will interpret the SVM and Random Forest Classification models to gain insights into the factors that influence diabetes prediction. We will also discuss the limitations of the proposed model and potential avenues for future research to further improve the accuracy and applicability of the model in real-world settings.

PROBLEM STATEMENT

The problem at hand is to develop a robust and accurate diabetes detection machine learning algorithm using Support Vector Machines (SVM) and Random Forest classification techniques. The algorithm will take in a dataset containing various health-related features of individuals, such as age, BMI, blood pressure, glucose levels, etc., and accurately predict whether an individual is diabetic or not.

This problem is of utmost importance as diabetes has become a global health epidemic, affecting millions of people worldwide. Early detection and management of diabetes can significantly improve health outcomes and prevent complications. However, accurate diagnosis of diabetes based on clinical parameters can be challenging, and machine learning algorithms have the potential to provide valuable insights and aid in the decision-making process.

The challenges associated with this problem statement include handling a diverse dataset with multiple features, addressing issues such as missing data, imbalanced data, and handling non-linearity in the data. Additionally, choosing the right machine learning algorithms, such as SVM and Random Forest, and optimising their hyper-parameters to achieve high accuracy, precision, recall, and F1-score are critical tasks in developing an effective diabetes detection algorithm.

The success of this project will be measured by achieving high accuracy and performance metrics on the test dataset, as well as validating the algorithm's robustness through cross-validation techniques. The developed algorithm will have the potential to be integrated into a real-world clinical setting to assist healthcare professionals in accurately detecting diabetes early and making informed treatment decisions.

SOFTWARE REQUIREMENT SPECIFICATIONS

- **Anaconda:**
  Anaconda Python is a free, open-source platform that allows you to write and execute code in the programming language Python. It is by continuum.io, a company that specialises in Python development. The Anaconda platform is the most popular way to learn and use Python for scientific computing, data science, and machine learning. It is used by over thirty million people worldwide and is available for Windows, macOS, and Linux.
  People like using Anaconda Python because it simplifies package deployment and management. It also comes with a large number of libraries/packages that you can use for your projects. Since Anaconda Python is free and open-source, anyone can contribute to its development.

- **Jupyter Notebook:**
  The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience. Jupyter supports over 40 programming languages, including Python, R, Julia, and Scala.

- Some Preinstalled libraries (seaborn,matplot, Pandas, Scikits Learn etc):

1. **NumPy:**
   It is used for working with arrays and linear algebra. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

2. **Pandas:**
   It is used for data analysis and data pre-processing, CSV file I/O (e.g. *pd.read_csv).* It is built on top of another package named Numpy, which provides support for multi-dimensional arrays.

3. **Sklearn:**

   It is used for making use of Machine learning tools. Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

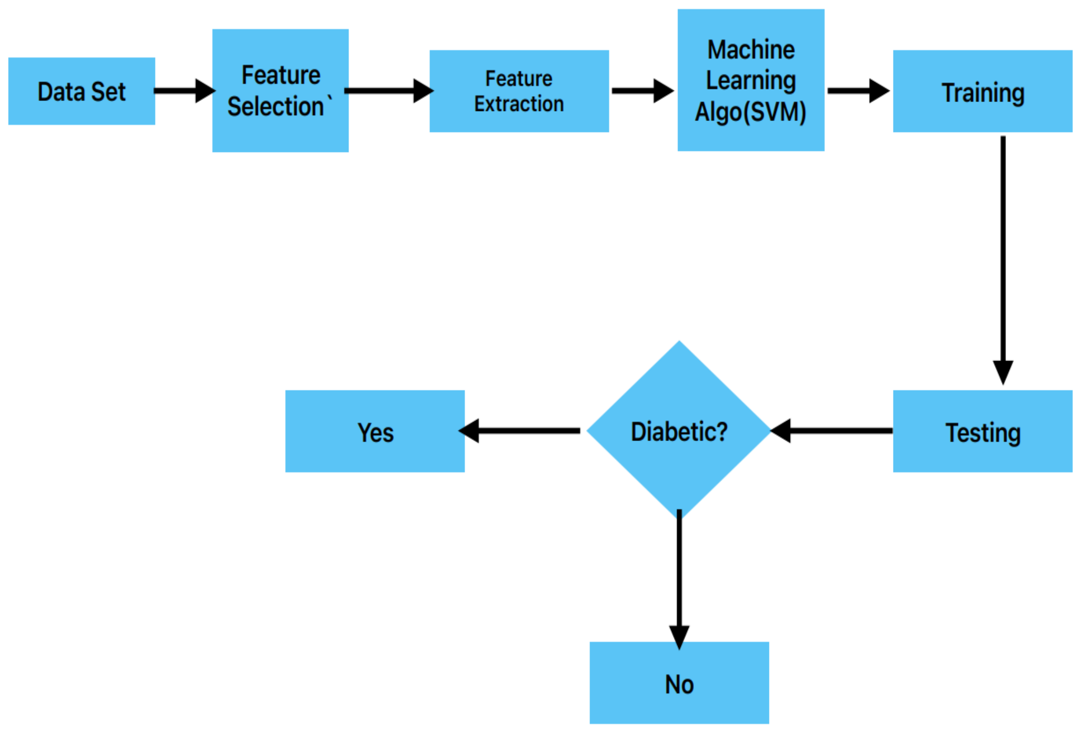
4. **Matplotlib:**

   It is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

5. **Seaborn:**

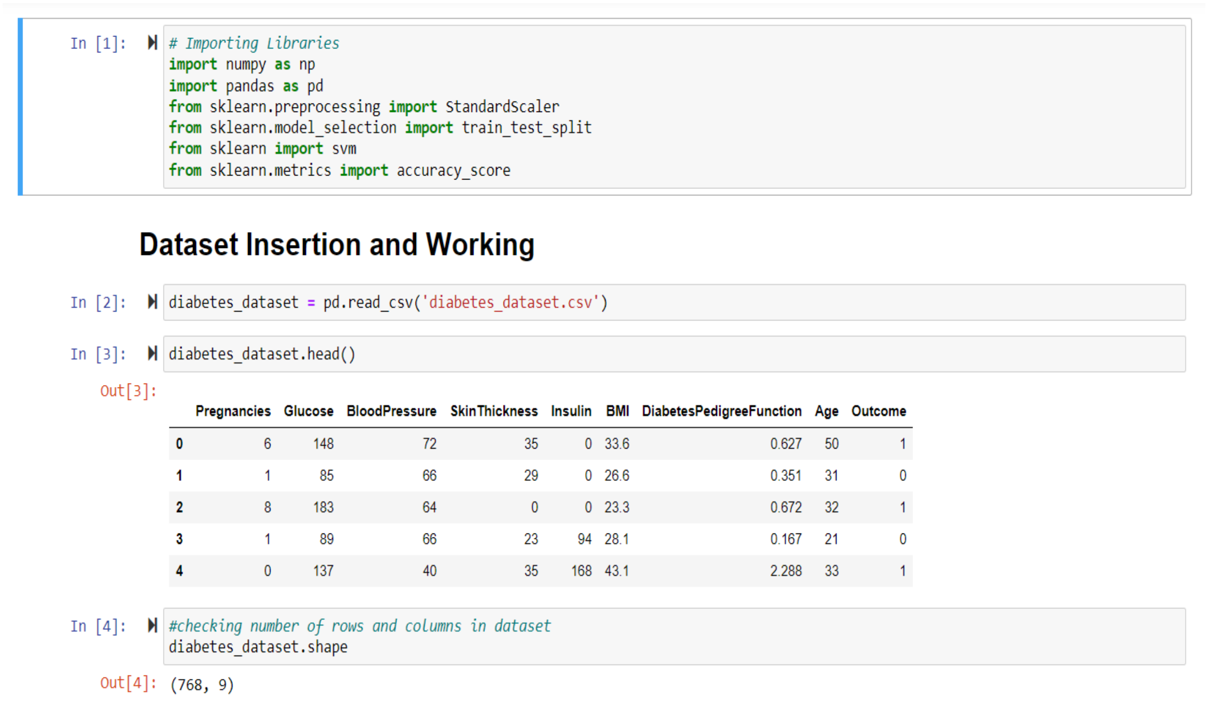
   Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. For a brief introduction to the ideas behind the library, you can read the introductory notes or the paper.

<u>ENITITY RELATIONSHIP DIAGRAM</u>



SNAPSHOTS OF THE DIFFERENT INPUT AND OUTPUT SCREENS

For SVM Model:



In [1]:
```
# Importing Libraries
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Dataset Insertion and Working

In [2]:
```
diabetes_dataset = pd.read_csv('diabetes_dataset.csv')
```

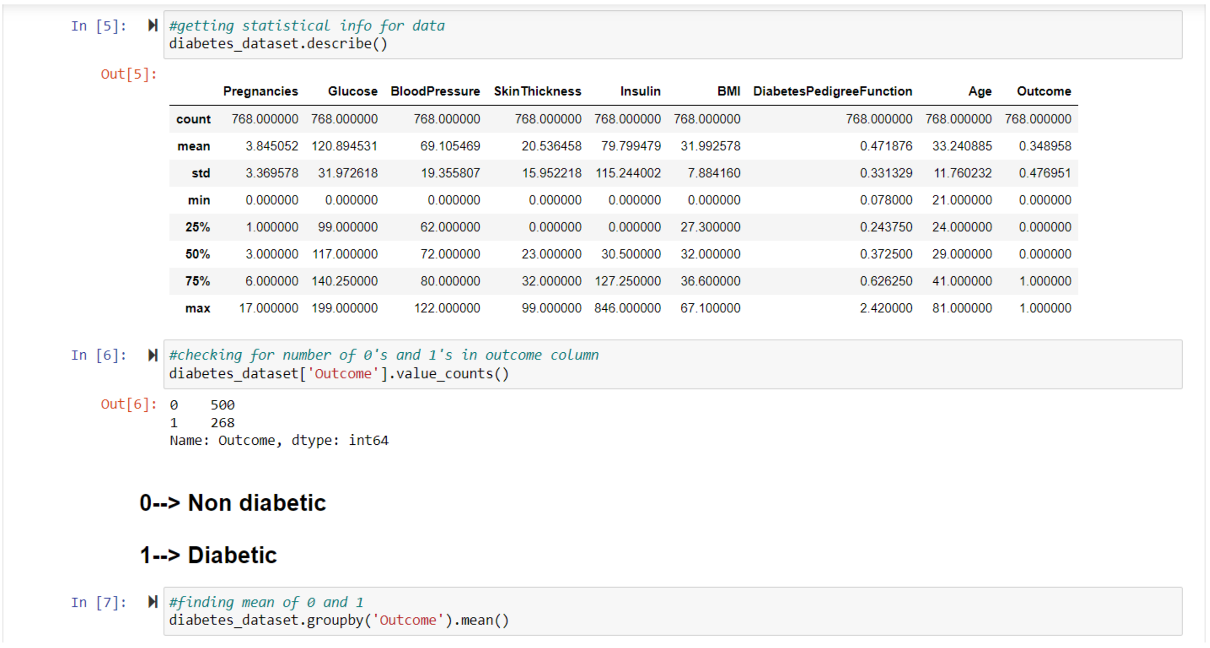
In [3]:
```
diabetes_dataset.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [4]:
```
#checking number of rows and columns in dataset
diabetes_dataset.shape
```

Out[4]:
```
(768, 9)
```



In [5]:
```
#getting statistical info for data
diabetes_dataset.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [6]:
```
#checking for number of 0's and 1's in outcome column
diabetes_dataset['Outcome'].value_counts()
```

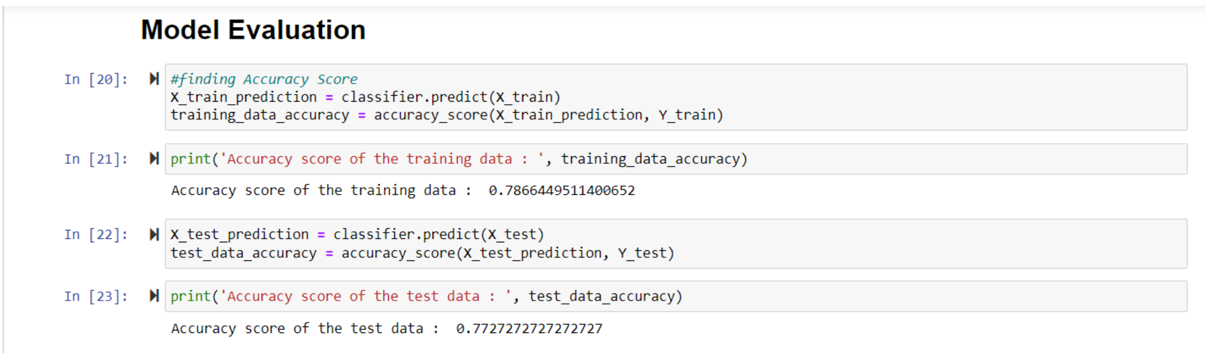
Out[6]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

0--> Non diabetic

1--> Diabetic

In [7]:
```
#finding mean of 0 and 1
diabetes_dataset.groupby('Outcome').mean()
```

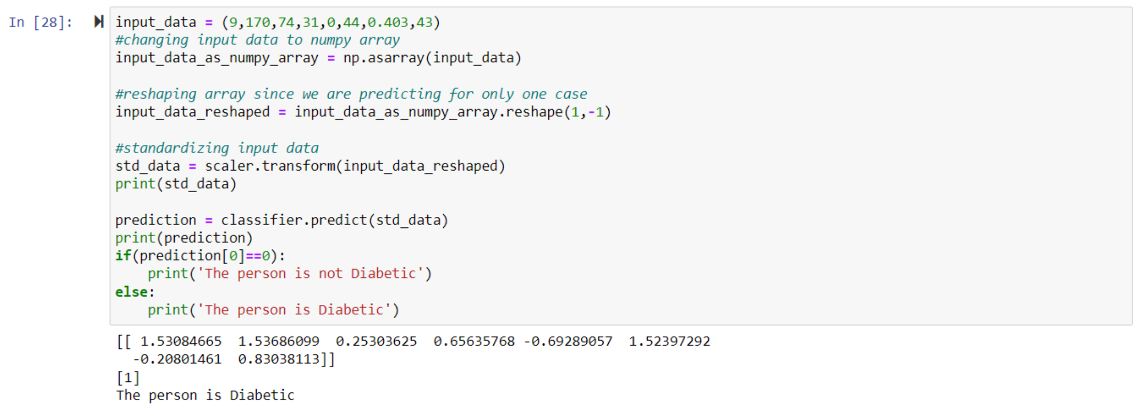
Model Evaluation(Checking accuracy):



Model Evaluation

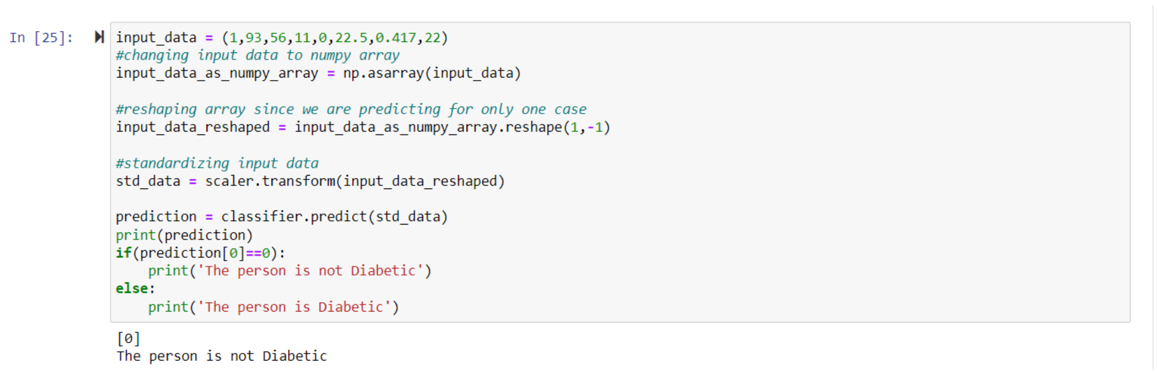
```
In [20]:  #finding Accuracy Score
          X_train_prediction = classifier.predict(X_train)
          training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

In [21]:  print('Accuracy score of the training data : ', training_data_accuracy)

          Accuracy score of the training data :  0.7866449511400652

In [22]:  X_test_prediction = classifier.predict(X_test)
          test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

In [23]:  print('Accuracy score of the test data : ', test_data_accuracy)

          Accuracy score of the test data :  0.7727272727272727
```

Positive Case of Diabetes:



```
In [28]:  input_data = (9,170,74,31,0,44,0.403,43)
          #changing input data to numpy array
          input_data_as_numpy_array = np.asarray(input_data)

          #reshaping array since we are predicting for only one case
          input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

          #standardizing input data
          std_data = scaler.transform(input_data_reshaped)
          print(std_data)

          prediction = classifier.predict(std_data)
          print(prediction)
          if(prediction[0]==0):
              print('The person is not Diabetic')
          else:
              print('The person is Diabetic')

          [[ 1.53084665  1.53686099  0.25303625  0.65635768 -0.69289057  1.52397292
            -0.20801461  0.83038113]]
          [1]
          The person is Diabetic
```

Negative Case of Diabetes:

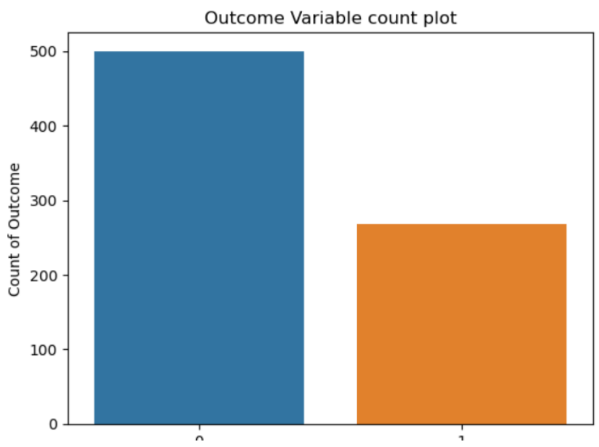


```
In [25]:  input_data = (1,93,56,11,0,22.5,0.417,22)
          #changing input data to numpy array
          input_data_as_numpy_array = np.asarray(input_data)

          #reshaping array since we are predicting for only one case
          input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

          #standardizing input data
          std_data = scaler.transform(input_data_reshaped)

          prediction = classifier.predict(std_data)
          print(prediction)
          if(prediction[0]==0):
              print('The person is not Diabetic')
          else:
              print('The person is Diabetic')

          [0]
          The person is not Diabetic
```

For Random Forest Classifier:

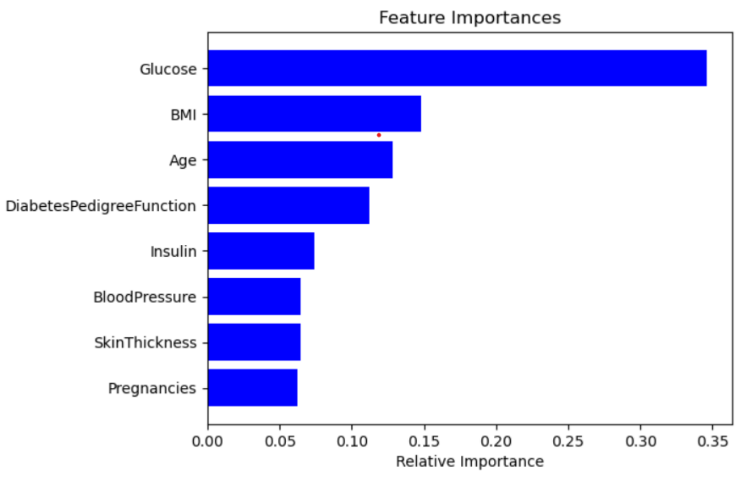
```
In [9]:  sns.countplot(diabetes_dataset['Outcome'])
         plt.xlabel("Outcome")
         plt.ylabel("Count of Outcome")
         plt.title("Outcome Variable count plot")
         plt.show()
```

```
C:\Users\adeeb ur rehman\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable a
s a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments withou
t an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Feature Importance

```
In [22]:  features= diabetes_dataset.columns
          importances=clf.feature_importances_
          indices=np.argsort(importances)
          plt.title('Feature Importances')
          plt.barh(range (len (indices)), importances [indices], color='b', align='center')
          plt.yticks (range (len(indices)), [features[i] for i in indices])
          plt.xlabel('Relative Importance')
          plt.show()
```



CONCLUSION

In conclusion, my machine learning project on diabetes using SVM and Random Forest classification has provided valuable insights and outcomes. Through the analysis and implementation of these two popular classification algorithms, I have learned several key points:

1. **Feature selection**: Selecting relevant features is crucial for improving the accuracy of machine learning models. I learned that careful feature selection, such as using domain knowledge and feature importance analysis, can significantly impact the performance of SVM and Random Forest classifiers for diabetes prediction.
2. **Model performance**: I gained an understanding of how SVM and Random Forest classifiers can be utilized for diabetes classification, and how their performance can be evaluated using metrics such as accuracy, precision, recall, and F1-score. I also learned how to use cross-validation techniques to ensure robust model evaluation and mitigate overfitting.
3. **Hyperparameter tuning**: I explored the importance of hyperparameter tuning for SVM and Random Forest classifiers. I learned how to optimize hyperparameters such as kernel type for SVM, and the number of trees, and feature sampling for Random Forest, to improve the model's performance.
4. **Ensemble methods**: I gained knowledge about the advantages of ensemble methods, specifically Random Forest, which combines multiple decision trees to improve classification accuracy and reduce overfitting.
5. **Interpretability**: I understood the limitations of SVM and Random Forest classifiers in terms of model interpretability. SVM produces a clear decision boundary, but may be less interpretable due to the use of kernel functions. Random Forest, on the other hand, provides feature importance rankings, which can help understand the contribution of each feature towards the prediction.

Overall, this project has helped me grasp the fundamental concepts of machine learning, including feature selection, model performance evaluation, hyperparameter tuning, ensemble methods, and model interpretability. It has also provided me with practical experience in implementing SVM and Random Forest classifiers for diabetes prediction. I can now confidently apply these concepts and techniques to future machine learning projects and continue to refine my skills in data analysis and predictive modeling.

LIMITATIONS

Machine learning is a powerful tool that can be used to analyze and make predictions from diabetes datasets.

However, like any other approach, it has its limitations. Here are some limitations of using machine learning on diabetes datasets:

1. **Limited Data Quality**: The quality of the diabetes dataset used for machine learning can significantly impact the accuracy and reliability of the results. If the dataset is incomplete, contains errors, or has missing values, it can lead to biased or inaccurate predictions.

2. **Lack of Data Diversity**: Machine learning algorithms require diverse and representative data to generalize well. If the diabetes dataset used for training is not diverse enough in terms of patient demographics, geographic locations, or other relevant factors, the model may not perform well on real-world data that differs from the training data.

3. **Feature Selection**: Selecting the most relevant features or variables from the diabetes dataset can be challenging. If important features are overlooked or irrelevant features are included, it can affect the model's performance and interpretability.

4. **Data Imbalance**: Diabetes datasets may be imbalanced, meaning that the distribution of positive (diabetic) and negative (non-diabetic) samples may be disproportionate. This can lead to biased model predictions, as the model may be more accurate in predicting the majority class and less accurate in predicting the minority class.
5. **Overfitting**: Machine learning models can be prone to overfitting, where the model learns to perform well on the training data but does not generalize well to new, unseen data. This can happen if

the model is too complex or if the dataset used for training is small, leading to a high risk of overfitting.

6. **Ethical Considerations**: Machine learning models for diabetes prediction or diagnosis may raise ethical concerns, such as privacy issues, fairness, and transparency. It is essential to ensure that the data used for training and the predictions made by the model are handled in a responsible and ethical manner.
7. **Causality vs. Correlation**: Machine learning algorithms are designed to identify patterns and correlations in data, but they may not always uncover causal relationships. Associations found in the diabetes dataset may not necessarily indicate causation, and care should be taken when interpreting the results.

8. **Changing Dynamics**: Diabetes is a complex and dynamic disease that can evolve over time. Machine learning models trained on a specific dataset may become less accurate as new data becomes available, requiring regular updates and monitoring to maintain their performance.
9. **Limited Interpretability**: Some machine learning algorithms, such as deep learning models, may lack interpretability, making it difficult to understand and explain the reasons behind their predictions. This can be a limitation when it comes to gaining trust and acceptance from stakeholders, including patients, clinicians, and regulatory authorities.

10. **Regulatory Requirements**: The use of machine learning in healthcare, including diabetes prediction or diagnosis, may be subject to regulatory requirements and compliance with laws, such as data protection and privacy regulations. Adhering to these requirements and obtaining necessary approvals can add complexity and limitations to the use of machine learning on diabetes datasets.

It is important to be aware of these limitations when using machine learning on diabetes datasets and to carefully evaluate the results and interpretations in a clinical context. Collaborating with healthcare professionals and domain experts can help mitigate some of these limitations and ensure that the machine learning models are used responsibly and effectively.

BIBLIOGRAPHY

- https://analyticsindiamag.com/understanding-the-basics-of-svm-with-example-and-python-implementation/
- https://www.datacamp.com/tutorial/random-forests-classifier-python
- https://www.kaggle.com/datasets/mathchi/diabetes-data-set
- https://stackoverflow.com/
- https://docs.streamlit.io/
- https://scikit-learn.org/0.21/documentation.html