

Diabetes Prediction

A Project Report

FALL SEM 2022-23

Submitted by

Puvvada Sai Dhanush (20BCI7319)

Guided By

Prof.Radha Mohan Pattanayak



VIT-AP UNIVERSITY

AMARAVATI

ANDHRA PRADESH, INDIA

2021-2022

Title: Predict Diabetes using Machine Learning.

Abstract:

In this project, our objective is to predict whether the patient has diabetes or not based on various features like *Glucose level, Insulin, Age, and BMI*. We will perform all the steps from *Data gathering to Model deployment*. During Model evaluation, we compare various machine learning algorithms on the basis of the accuracy_score metric and find the best one. Then we create a web app using Flask which is a python micro-framework.

Motivation:

International Diabetes Federation (IDF) stated that 382 million people are living with diabetes worldwide. Over the last few years, the impact of diabetes has increased drastically, which makes it a global threat. At present, Diabetes has steadily been listed in the top position as a major cause of death. The number of affected people will reach up to 629 million i.e. 48% increase by 2045. However, diabetes is largely preventable and can be avoided by making lifestyle changes. These changes can also lower the chances of developing heart disease and cancer. So, there is a dire need for a prognosis tool that can help the doctors with early detection of the disease and hence can recommend the lifestyle changes required to stop the progression of the deadly disease. Diabetes is an increasingly growing health issue due to our inactive lifestyle. If it is detected in time then through proper medical treatment, adverse effects can be prevented. To help in early detection, technology can be used very reliably and efficiently. Using machine learning we have built a predictive model that can predict whether the patient is diabetes positive or not.

Keywords :

Diabetes, Prediction, Accuracy

SVM

Logistic Regression

Decision Tree

Random forest

Gaussian NB

KNN

XG Boost

Existing work:

P.Yasodha and N.R.Ananthanarayanan ,in their research paper they have compared classification algorithm using WEKA tool on clinical available dataset. In that they have employed J48, LAD Tree, REP Tree and estimated the accuracy of algorithms when the attributes have changed. Shelly Gupta, Dharminder Kumar and Anand Sharma , they have been employed various classifiers using WEKA tool on PIMA Indian Diabetes, Wisconsin Breast cancer, StatLog Heart Disease and BUPA Liver Disorder Health care datasets and discovered all the classification tools were produce same accuracy and times to build a model or training set on same dataset.

When the dataset and attributes changes it behaves differently on every datasets and produce various accuracy rates and times. Raj Kumar and Dr. Rajesh Verma, presented a paper on classification algorithms for datamining as a survey to understand how the classification algorithm behaves differently on various scenarios. Jianchao han , in his research work the decision tree using WEKA has been used to build the prediction model of the type2 diabetes data set. These models consider the Plasma Insulin attribute as the main attribute for predicting the disease

Proposed work:

Diabetes prediction using machine learning classification algorithms.

Flow charts:

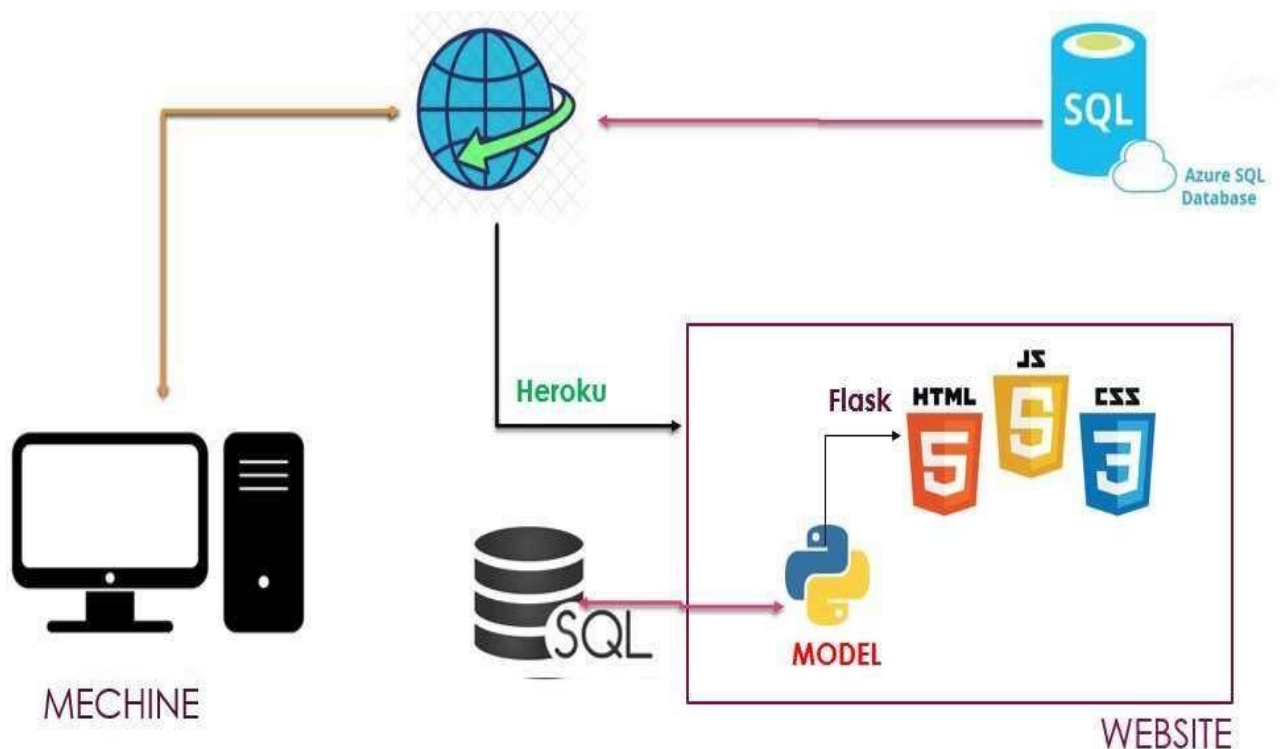
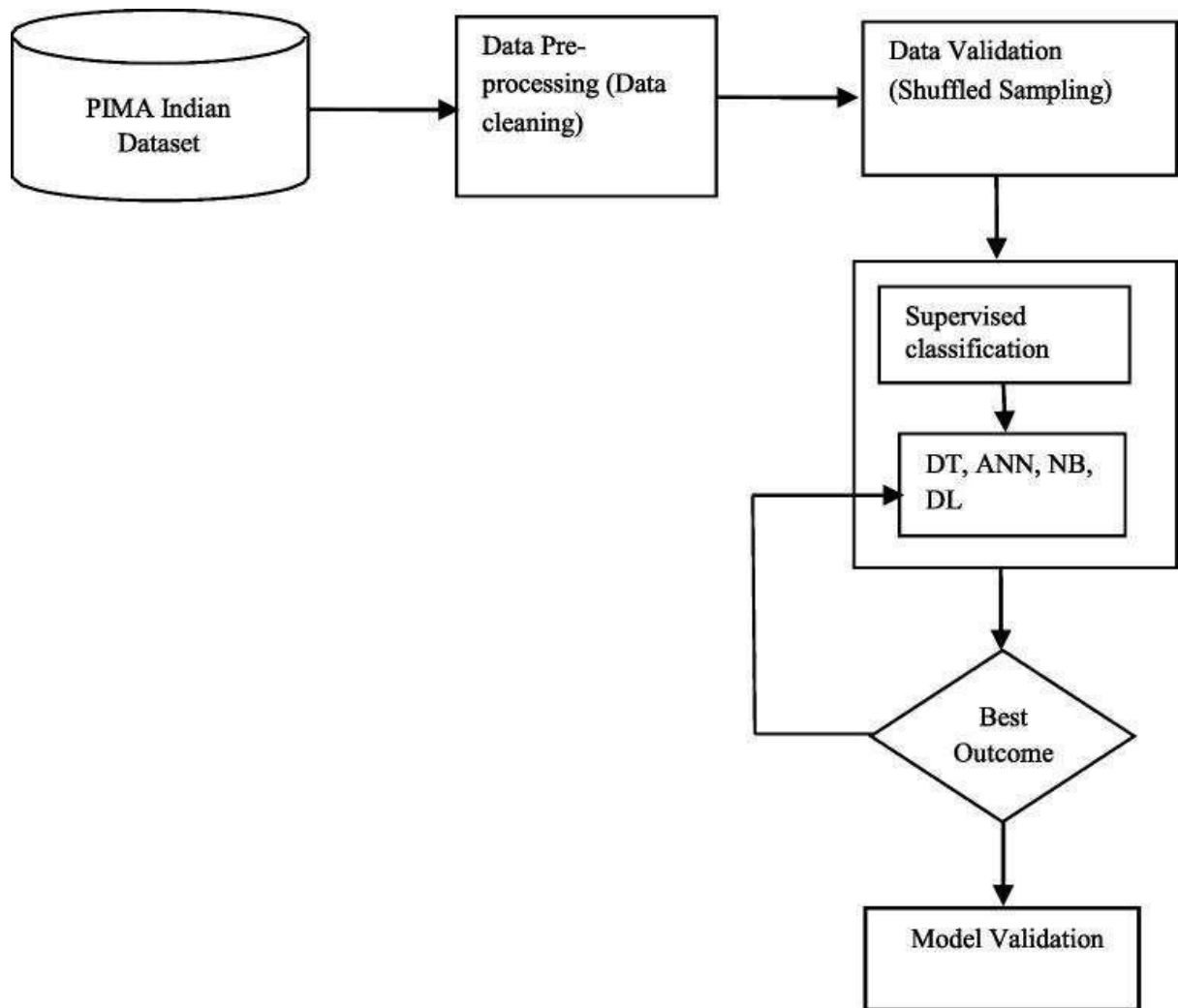
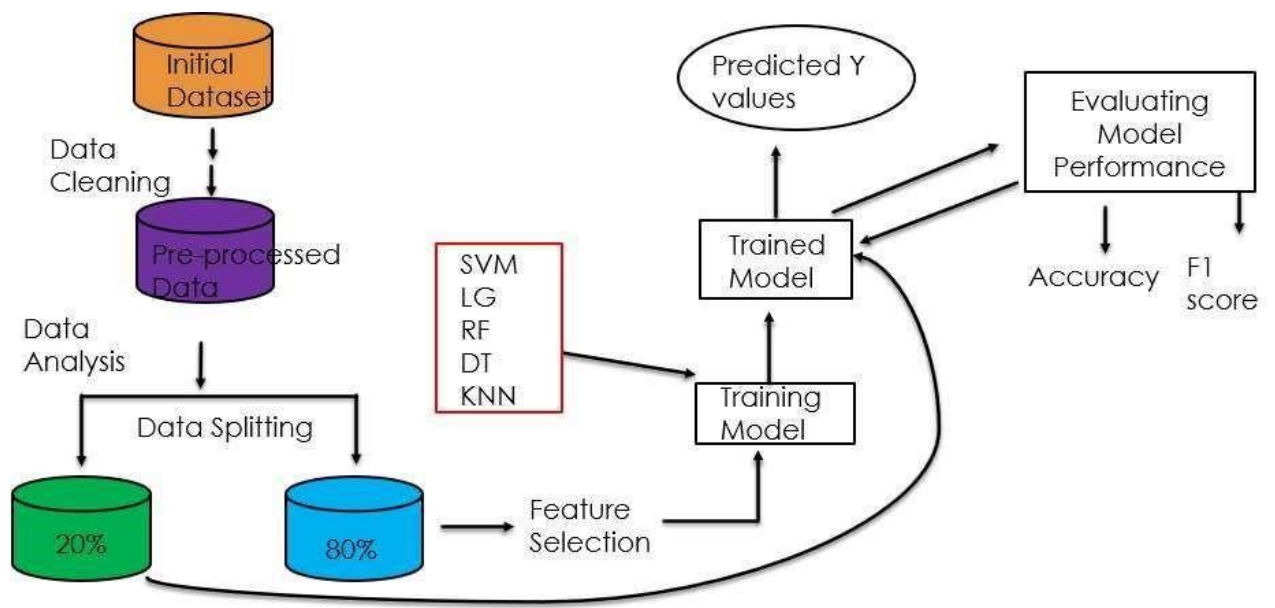


Fig:1 Deployment flowchart



Code:

<https://github.com/Dhanushpuvvada/Diabetes-Prediction/>

Introduction:

Considering the current scenario, in developing countries like India, Diabetes has become a very severe disease. Around 425 million people suffer from diabetes. Approximately 2-5 million patients every year lose their lives due to diabetes. It is said that by 2045 this will rise to 629 million.

Diabetes is classified as-

Type-1 is known as Insulin-Dependent Diabetes Mellitus (IDDM). The inability of the human body to generate sufficient insulin is the reason behind this type of Diabetes and hence it is required to inject insulin into a patient.

Type-2 is also known as Non-Insulin-Dependent Diabetes Mellitus (NIDDM). This type of Diabetes is seen when body cells are not able to use insulin properly.

Type-3 Gestational Diabetes, an increase in blood sugar level in a pregnant woman where diabetes is not detected earlier results in this type of diabetes.

A technique called, Predictive Analysis, incorporates a variety of machine learning algorithms, data mining techniques, and statistical methods that uses current and past data to find knowledge and predict future events. By applying predictive analysis to healthcare data, significant decisions can be taken and predictions can be made. Predictive analytics can be done using machine learning algorithms. Predictive analytic aims at diagnosing the disease with the best possible accuracy, enhancing patient care, optimizing resources along with improving clinical outcomes. Machine learning is considered to be one of the most important artificial intelligence features that support the development of computer systems having the ability to acquire knowledge from past experiences with no need for programming for every case. Machine learning is considered to be a dire need in today's situation to eliminate human efforts by supporting automation with minimum flaws. The Existing method for diabetes detection uses lab tests such as fasting blood glucose and oral glucose tolerance.

About the data: This dataset has various features like *Glucose level, Insulin, Age, and BMI*.

Data:

<https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>

Machine Specifications:

Table of Content	Description
Operating system	Windows 10 & Windows 11 64-bit operating system
RAM	8GB or 16GB
GRAPHIS	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Editors	Jupyter Notebook, Spyder, Google Colab

Requirements:

Dependencies include python libraries like

```
``` sklearn  

matplotlib

pandas

seaborn

```
```

Classification models used

```
```  

Decision Trees

Random forest

Logistic Regression

SVM

Naive Bayes

Logistic Regression

Ensemble Modeling

```
```

Methodology:

Step 1:

Data gathering and Importing libraries.

All the standard libraries like NumPy, pandas, matplotlib and seaborn are imported in this step. We use NumPy for linear algebra operations, pandas for using data frames, and matplotlib and seaborn for plotting graphs. The dataset is imported using the pandas `read_csv()`.

```
#Importing dataset
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from warnings import filterwarnings
filterwarnings(action='ignore')
```

Loading data set

```
data = pd.read_csv("diabetes.csv")
print("Successfully Imported Data!")
output:
Successfully Imported Data!
```

Step 2:

Descriptive Analysis

```
# Preview data  
dataset.head()
```

 Output:

| | 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 |

```
print(data.shape)
```

 output:

(768, 9)

Description

```
data.describe(include='all')
```

| | 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 | 3.845062 | 120.894531 | 69.105489 | 20.536458 | 79.799479 | 31.992578 | 0.471876 | 33.240885 | 0.348968 |
| std | 3.369578 | 31.972018 | 19.355807 | 15.952218 | 115.244002 | 7.694160 | 0.331329 | 11.760232 | 0.478951 |
| 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 |

Finding Null Values

```
print(data.isna().sum())
```

```
Pregnancies      0
Glucose           0
BloodPressure     0
SkinThickness     0
Insulin           0
BMI               0
DiabetesPedigreeFunction  0
Age              0
Outcome           0
dtype: int64
```

- There are a total of 768 records and 9 features in the dataset.
- Each feature can be either of integer or float data type.
- Some features like Glucose, Blood pressure, Insulin, BMI have zero values which represent missing data.
- There are zero NaN values in the dataset.
- In the outcome column, 1 represents diabetes positive and 0 represents diabetes negative

```
data.corr()
```

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|--------------------------|-------------|----------|---------------|---------------|-----------|----------|--------------------------|-----------|----------|
| Pregnancies | 1.000000 | 0.129459 | 0.141282 | -0.081672 | -0.073535 | 0.017683 | -0.033523 | 0.544341 | 0.221898 |
| Glucose | 0.129459 | 1.000000 | 0.152590 | 0.057328 | 0.331357 | 0.221071 | 0.137337 | 0.263514 | 0.466581 |
| BloodPressure | 0.141282 | 0.152590 | 1.000000 | 0.207371 | 0.088933 | 0.281805 | 0.041265 | 0.239528 | 0.065068 |
| SkinThickness | -0.081672 | 0.057328 | 0.207371 | 1.000000 | 0.436783 | 0.392573 | 0.183928 | -0.113970 | 0.074752 |
| Insulin | -0.073535 | 0.331357 | 0.088933 | 0.436783 | 1.000000 | 0.197859 | 0.185071 | -0.042163 | 0.130548 |
| BMI | 0.017683 | 0.221071 | 0.281805 | 0.392573 | 0.197859 | 1.000000 | 0.140647 | 0.036242 | 0.292695 |
| DiabetesPedigreeFunction | -0.033523 | 0.137337 | 0.041265 | 0.183928 | 0.185071 | 0.140647 | 1.000000 | 0.033561 | 0.173844 |
| Age | 0.544341 | 0.263514 | 0.239528 | -0.113970 | -0.042163 | 0.036242 | 0.033561 | 1.000000 | 0.238358 |
| Outcome | 0.221898 | 0.466581 | 0.065068 | 0.074752 | 0.130548 | 0.292695 | 0.173844 | 0.238358 | 1.000000 |

```
data['Outcome'].value_counts()
```

```
0    500
1    268
Name: Outcome, dtype: int64
```

0 means no DIABETED

1 means patient with DIABETED

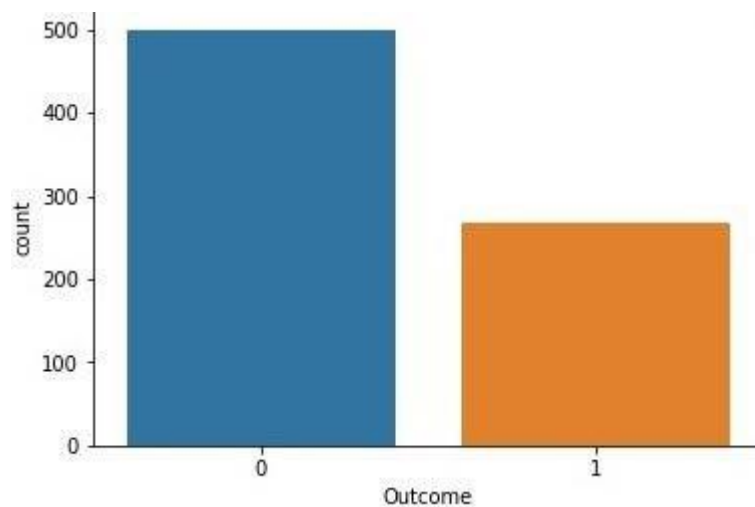
No.of 0 in dataset is 500

No.of 1 in the dataset is 268

Step 3:

Data Visualizations

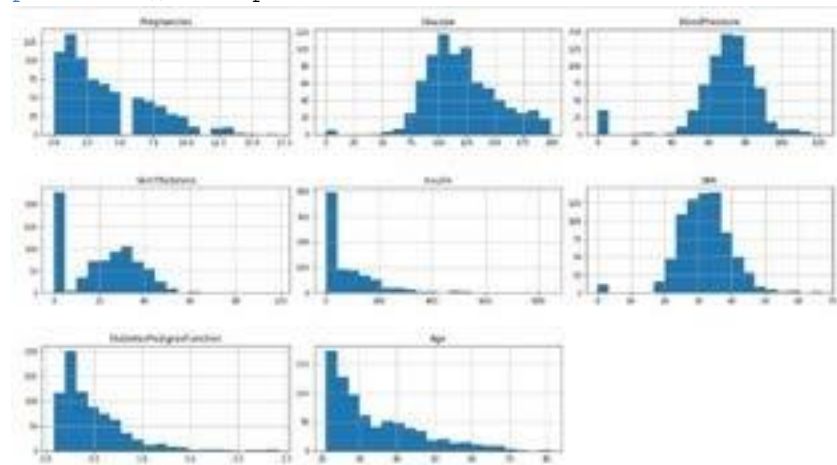
```
# Outcome counterplot
sns.countplot(x = 'Outcome', data = dataset)
```



Outcome Countplot

```
# Histogram of each feature
```

```
import itertools
col = dataset.columns[8:]
plt.subplots(figsize = (20, 15))
length = len(col)
for i, j in itertools.zip_longest(col, range(length)):
    plt.subplot((length/2), 3, j + 1)
    plt.subplots_adjust(wspace = 0.1, hspace = 0.5)
    dataset[i].hist(bins = 20)
    plt.title(i)
plt.show()
output:
```



Histogram of each Feature

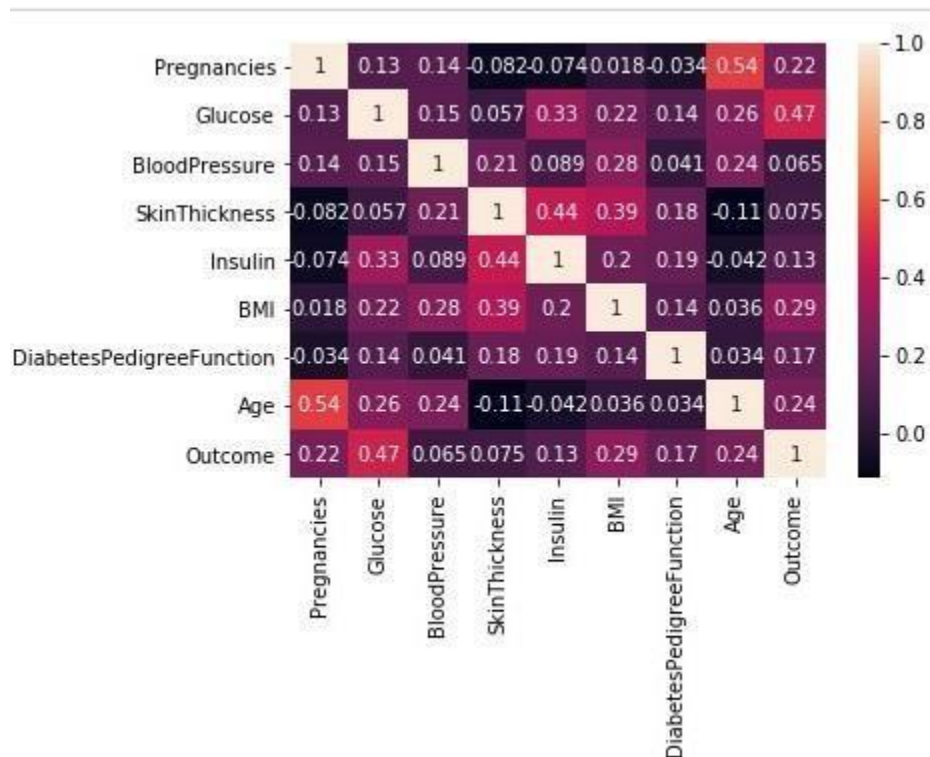
```
# Pairplot
sns.pairplot(data = dataset, hue = 'Outcome')
plt.show()
Output:
```



Pairplot of all features

```
# Heatmap
sns.heatmap(dataset.corr(), annot = True)
plt.show()
```

Output:



Heatmap of Feature correlation

- The counterplot tells us that the dataset is imbalanced, as the
- From the correlation heatmap, we can see that there is a high correlation between Outcome and [Glucose, BMI, Age, Insulin]. We can select these features to accept input from the user and predict the outcome.

Note: There are so many other plotting you can refer to on [Github](#).

Feature Selection

```
#Lets extract features and targets
X = data.drop(columns=['Outcome'])
Y = data['Outcome']
print("Features Extraction Sucessfull")
```

Features Extraction Sucessfull

Feature Importance

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

```
from sklearn.ensemble import ExtraTreesClassifier
classifiern = ExtraTreesClassifier()
classifiern.fit(X,Y)
score = classifiern.feature_importances_
print(score)
```

```
[0.10925873 0.22667986 0.0945291  0.07962772 0.07630627 0.14638875
 0.12027243 0.14693715]
```

Splitting Dataset

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2)
```


Step 4:

Applying classification algorithms

Using Logistic Regression

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,Y_train)
Y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score,confusion_matrix
print("Accuracy Score:",accuracy_score(Y_test,Y_pred))
```

Accuracy Score: 0.8181818181818182

Using KNN

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
```

Accuracy Score: 0.7272727272727273

Using SVC

```
from sklearn.svm import SVC
model = SVC()
model.fit(X_train,Y_train)
pred_y = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,pred_y))
```

Accuracy Score: 0.8116883116883117

Using Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='entropy',random_state=7)
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
```

Accuracy Score: 0.7012987012987013

Using GaussianNB

```
from sklearn.naive_bayes import GaussianNB
model3 = GaussianNB()
model3.fit(X_train,Y_train)
y_pred3 = model3.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred3))
```

Accuracy Score: 0.8051948051948052

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
model2 = RandomForestClassifier(random_state=1)
model2.fit(X_train, Y_train)
y_pred2 = model2.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred2))
```

Accuracy Score: 0.7922077922077922

Using Xgboost

```
import xgboost as xgb
model5 = xgb.XGBClassifier(random_state=1)
model5.fit(X_train, Y_train)
y_pred5 = model5.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred5))
```

Accuracy Score: 0.7597402597402597

Comparing the accuracy:

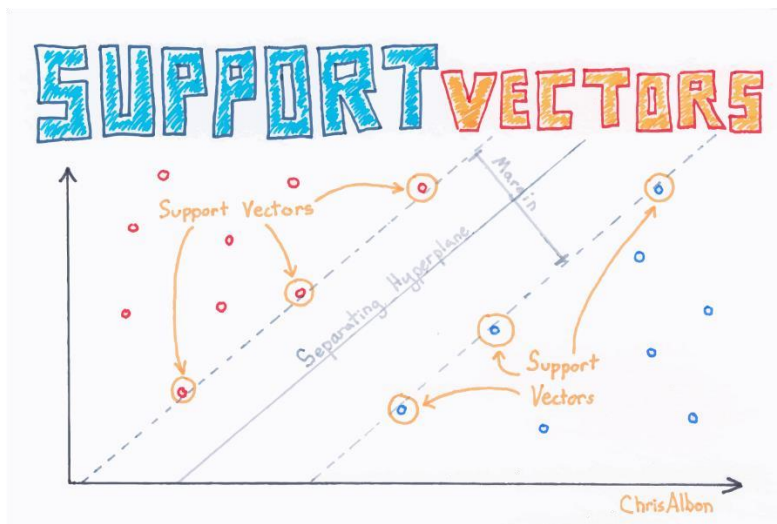
Results

```
results = pd.DataFrame({
    'Model': ['Logistic Regression', 'KNN', 'SVC', 'Decision Tree', 'GaussianNB', 'Random Forest', 'Xgboost'],
    'Score': [.81818181818182, 0.72727272727273, 0.8196883116883117, 0.8051948051948052, 0.733, 0.793, 0.750]})

result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
print(result_df)
```

| Score | Model |
|----------|---------------------|
| 0.819688 | SVC |
| 0.818182 | Logistic Regression |
| 0.805195 | Decision Tree |
| 0.793000 | Random Forest |
| 0.750000 | Xgboost |
| 0.733000 | GaussianNB |
| 0.727273 | KNN |

The Algorithm selected for deployment:



Support Vector Classifier(SVC) is a type of supervised classification model whose objective is to classify the data based on a maximal margin hyperplane build using support vectors. This hyperplane is a decision boundary that classifies various classes. It is built using support vectors, which are the outliers. The hyperplane which has the highest margin is selected as the decision boundary.

SVCs can classify linear as well as non-linear data using a kernel trick that implicitly maps the input to highdimensional vector spaces. This kernel trick converts the lower-dimensional feature space into higherdimensional feature space which is linearly separable. For example, data in a 2D may not be linearly separable but when it is converted into 3D using the kernel function it becomes linearly separable.

SVC has three main parameters that affect the performance of the model which are the kernel, gamma, and C. The kernel parameter signifies the type of kernel which can be "Linear" for linearly separable data or "RBF", or "poly" for non-linearly separable data. The Gamma parameter is the kernel coefficient. As the value of gamma increases, it tries to exactly fit the dataset which gives generalization error and causes overfitting. C parameter is the cost of misclassification of the model. The high value of C gives you low bias and high variance whereas the low value of C gives you high bias and low variance.

Step 5:

Model Evaluation and Flask code

```
# Evaluating using accuracy_score metric from  
sklearn.metrics import accuracy_score  
accuracy = accuracy_score(Y_test, Y_pred)print("Accuracy: " + str(accuracy *  
100))
```

Output:

Accuracy: 73.37662337662337# Confusion matrix

```
from sklearn.metrics import confusion_matrix  
cm = confusion_matrix(Y_test, Y_pred) cm
```

Output: array([[87, 13],
[20, 34]], dtype=int64)

```
# Heatmap of Confusion matrix  
sns.heatmap(pd.DataFrame(cm), annot=True)  
Output:
```



Heatmap of Confusion Matrix

```
# Classification report
```

```

from sklearn.metrics import classification_report
print(classification_report(Y_test, Y_pred))

```

Output:

| | | precision | recall | f1-score | |
|-----------|------|-----------|--------|----------|------------------|
| support | | | | | |
| | 0.0 | 0.81 | 0.87 | 0.84 | 100 |
| | 1.0 | 0.72 | 0.63 | 0.67 | 54 |
| micro avg | | 0.79 | 0.79 | 0.79 | 154 |
| avg | 0.77 | 0.75 | 0.76 | | 154 weighted avg |
| 0.78 | 0.79 | 0.78 | 154 | | |

We have chosen three metrics accuracy_score, confusion matrix, and classification report for evaluating our model.

Flask code:

```

import numpy as np import pandas as pd from flask import
Flask, request, jsonify, render_template import pickle

app = Flask(__name__) model =
pickle.load(open('model.pkl', 'rb'))

dataset =
pd.read_csv('diabetes.csv')

dataset_X = dataset.iloc[:, [1, 2, 5, 7]].values

from sklearn.preprocessing import
MinMaxScaler sc = MinMaxScaler(feature_range =
(0,1)) dataset_scaled =
sc.fit_transform(dataset_X) @app.route('/')

def home():

    return render_template('index.html')

@app.route('/predict',methods=['POST']) def
predict():

```

```

'''
For rendering results on HTML GUI
'''
float_features = [float(x) for x in
request.form.values()]
final_features =
[np.array(float_features)]
prediction = model.predict(
sc.transform(final_features) )

if prediction ==
1:

    pred = "You have Diabetes, please consult a Doctor."
elif prediction == 0:

    pred = "You don't have Diabetes."

output = pred

return render_template('index.html',
prediction_text='{}'.format(output))

if __name__ ==

"__main__":

app.run(debug=True)

```

Step 6:

Model Deployment

<http://127.0.0.1:5000/>



For input [150,0,35,45]

Output:

You have Diabetes, please consult a Doctor.

Results:

Machine learning algorithm report-

| Model | score |
|---------------------|-------|
| SVC | 0.81 |
| Logistic Regression | 0.81 |

| | |
|---------------|-------------|
| Decision Tree | 0.80 |
| Random Forest | 0.75 |
| GaussianNB | 0.73 |
| KNN | 0.72 |
| XGBoost | 0.74 |

SVC classification report-

| | precision | recall | f1-score | support |
|---------------------|-----------|--------|----------|---------|
| 0.0 | 0.81 | 0.87 | 0.84 | 100 |
| 1.0 | 0.72 | 0.63 | 0.67 | 54 |
| micro avg | 0.79 | 0.79 | 0.79 | 154 |
| macro avg | 0.77 | 0.75 | 0.76 | 154 |
| weighted avg | 0.78 | 0.79 | 0.78 | 154 |

Conclusion:

In this project, seven machine learning classification methods were implemented, and their results were compared with different statistical measures. Various machine learning algorithms are applied to the data set and the classification has been done using various algorithms of which Logistic Regression and support vector machine gives the highest accuracy of 81%. It is clear that the model improves the accuracy and precision of diabetes prediction with this data set. Further, this work can be extended to find how likely nondiabetic people can have diabetes in the next few years.