

AI Project Algorithm Report



Session: 2021 – 2025

Submitted To:

Mr. Samyan Qayyum Wahla

Submitted By:

Subhan Anjum

2021-CS-13

Department of Computer Science
University of Engineering and Technology Lahore Pakistan

Contents

	Diabetes Detection Using KNN Algorithm: A Predictive Modeling Approach	1
1	Introduction	1
2	Background	1
3	Objective	1
4	Code Appendix	1
4.1	Importing libraries	1
4.2	Loading the Dataset	1
4.3	Replace columns like [Glucose,BloodPressure,SkinThickness,BMI,Insulin] with Zero as values with mean of respective column	2
4.4	Extracting independent variables	2
4.5	Extracting dependent variable	2
4.6	Exploring data to know relation before processing	2
4.7	Splitting dataset into training and testing set	3
4.8	Feature scaling	4
4.9	Loading model - KNN	4
4.10	Fitting model	4
4.11	Making predictions	4
4.12	Evaluating model	4
4.13	Accuracy	4
5	Conclusion	4

Diabetes Detection Using KNN Algorithm: A Predictive Modeling Approach

1 Introduction

Diabetes is a prevalent and chronic health condition with significant global impact. Early detection and intervention are crucial for effective management and improved patient outcomes. This report delves into the application of the k-Nearest Neighbors (KNN) algorithm, a powerful supervised learning technique, to predict the likelihood of diabetes in individuals based on key health indicators.

2 Background

The rising incidence of diabetes underscores the need for innovative approaches to enhance early diagnosis. Machine learning algorithms, particularly KNN, have demonstrated effectiveness in healthcare applications. KNN is well-suited for classification tasks, making it a suitable candidate for predicting diabetes based on relevant health data.

3 Objective

The primary goal of this project is to develop a predictive model using the KNN algorithm to assess the probability of diabetes in individuals. By leveraging data such as glucose levels, blood pressure, skin thickness, BMI, and insulin levels, the model aims to provide accurate and timely predictions for better patient outcomes.

4 Code Appendix

A section containing key excerpts of the Python code used in the project for reference. This includes data loading, preprocessing steps, model training, and evaluation.

4.1 Importing libraries

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

4.2 Loading the Dataset

```
data = pd.read_csv('diabetes.csv')
data.head()
```

	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

4.3 Replace columns like [Glucose,BloodPressure,SkinThickness,BMI,Insulin] with Zero as values with mean of respective column

```
zero_not_accepted = ['Glucose','BloodPressure','SkinThickness','BMI','Insulin']
# for col in zero_not_accepted:
#     for i in data[col]:
#         if i==0:
#             colSum = sum(data[col])
#             meanCol=colSum/len(data[col])
#             data[col]=meanCol

for col in zero_not_accepted:
    data[col]= data[col].replace(0,np.NaN)
    mean = int(data[col].mean(skipna=True))
    data[col] = data[col].replace(np.NaN,mean)
```

4.4 Extracting independent variables

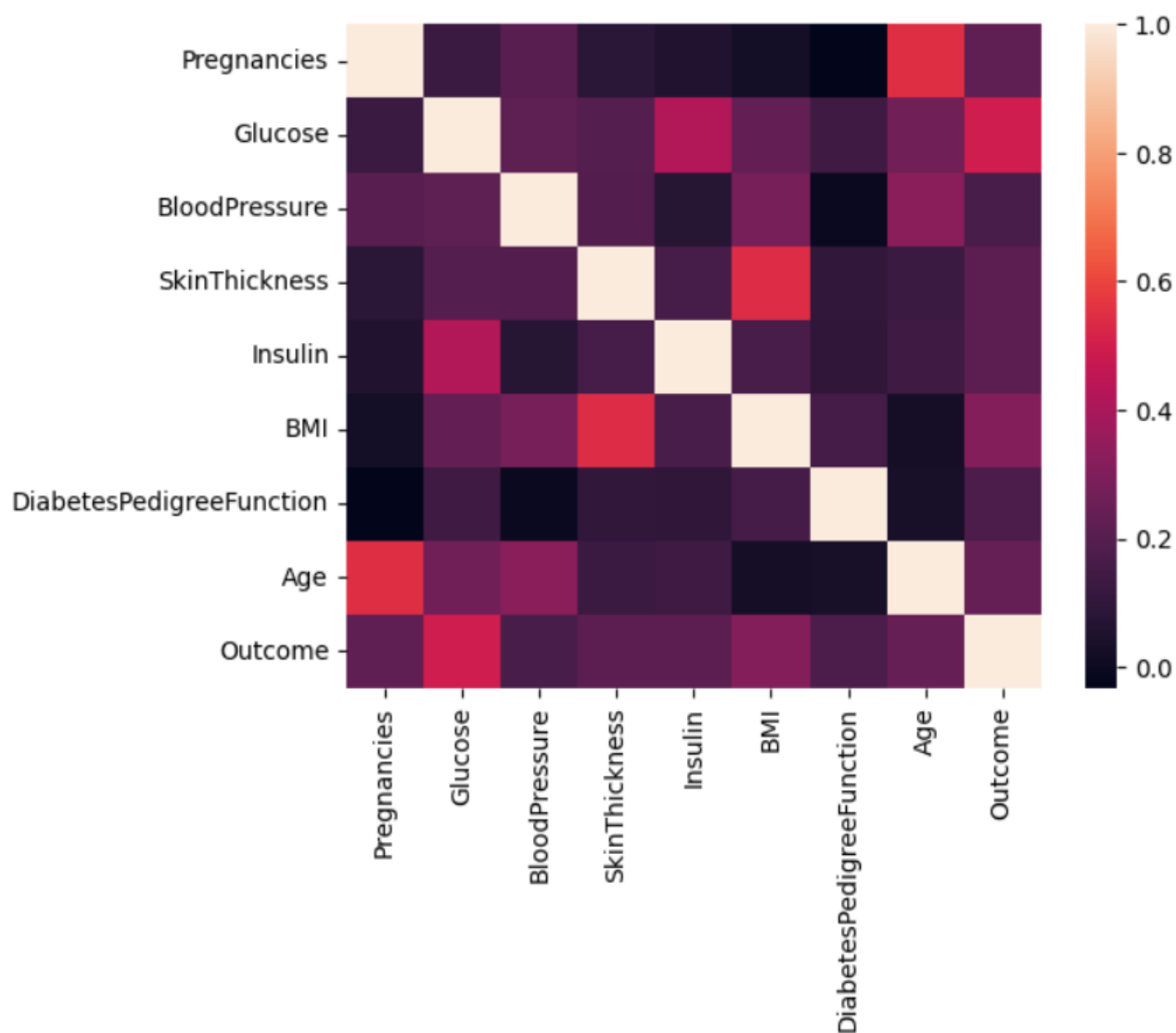
```
X = data.iloc[:,0:8]
```

4.5 Extracting dependent variable

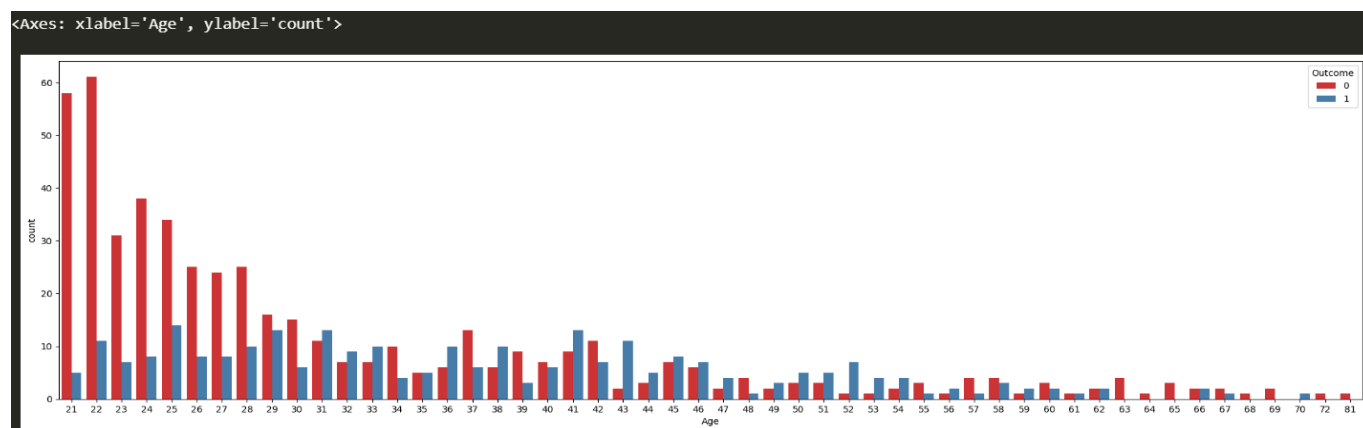
```
y = data.iloc[:,8]
```

4.6 Exploring data to know relation before processing

```
sns.heatmap(data.corr())
```



```
plt.figure(figsize=(25,7))
sns.countplot(x='Age',hue='Outcome',data=data,palette='Set1')
```



4.7 Splitting dataset into training and testing set

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

4.8 Feature scaling

```
scaler = StandardScaler()  
X_train = scaler.fit_transform(X_train)  
X_test = scaler.transform(X_test)
```

4.9 Loading model - KNN

```
classifier = KNeighborsClassifier(n_neighbors=11,p=2,metric='euclidean')
```

4.10 Fitting model

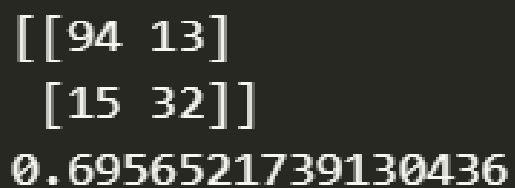
```
classifier.fit(X_train,y_train)
```

4.11 Making predictions

```
y_pred = classifier.predict(X_test)
```

4.12 Evaluating model

```
conf_matrix = confusion_matrix(y_test,y_pred)  
print(conf_matrix)  
print(f1_score(y_test,y_pred))
```



```
[[94 13]  
 [15 32]]  
0.6956521739130436
```

4.13 Accuracy

```
print(accuracy_score(y_test,y_pred))
```



```
0.8181818181818182
```

5 Conclusion

In conclusion, the application of the KNN algorithm for diabetes prediction showcases promising results. The model's ability to leverage key health indicators underscores the potential of machine learning in improving healthcare outcomes. However, it is essential to acknowledge the limitations and consider further refinements to enhance the model's robustness. Future work may involve [insert potential improvements or extensions].