“DIABETES PREDICTION BY K-NN”

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

RGUKT

Rajiv Gandhi University of Knowledge

TechnologiesR.K.VALLEY

Submitted by

A.G.DHARANI-R170672

Under the Esteemed guidance of

Mr. SatyaNandaram
NRGUKT RK Valley.

DECLARATION

We hereby declare that the report of the Bachelor Of Technology Minor Project Work entitled **“DIABETES PREDICTION”** which is being submitted to Rajiv Gandhi University of Knowledge Technologies, RK Valley, in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology in Computer Science and Engineering, is a bona-fide report of the work carried out by us.

A.G.DHARANI-R170672
Dept. Of Computer
Science and Engineering

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

RGUKT

(A.P. Government Act 18 of 2008) RGUKT, RK VALLEY
Department of Computer Science and Engineering

CERTIFICATE FOR PROJECT COMPLETION

This is certify that the project entitled "DIABETES PREDICTION BY KNN" submitted by **A.G.DHARANI** of ID NO:**R170672** under our guidance and supervision for the partial fulfillment for the degree Bachelor of Technology in ComputerScience and Engineering during the Academic semester-2 2021-2022 at RGUKT, RK VALLEY. To the best of my knowledge, the results embodied in thisdissertation work have not been submitted to any University or Institute for the award of any degree or diploma.

Project Internal Guide	**Head of the Department**
Mr.N.SatyaNandaram	Mr. P.Harinadha
Assistant Professor	HOD Of CSE
RGUKT, RK Valley	RGUKT, RK Valley

Abstract

Diabetes is the most common disease worldwide and keeps increasing everyday due to changing lifestyles, unhealthy food habits and overweight problems. We normally know predicting the diabetes through physical and chemical tests, are available for diagnosing diabetes. Data Science methods have the potential of predicting diabetes.

The study was carried out to find the significance of health-related predictors of diabetes in **Pima Indians Women**. The study population was the females 21 years and above of Pima Indian heritage patients of diabetes and digestive and kidney diseases. why so many Pima Indian Women suffer from diabetes in relation to other ethnicities? The purpose of the study was to find out the factors that are associated with the presence of diabetes in Pima Indians. To find out the reason behind this, we have to first analyze the relationship between different features, such as the number of times a woman was pregnant, their BMI, prevalence of diabetes, etc.

Index

Introduction 6

Data Collection..6

Libraries ……………………………………………………………...7-9

Exploratory Data Analysis ………………………………………………10-19

Prediction Using K-NN…………………………………………………19-21

Result Visualisation……………………………………………………22

Model Performance Analysis………………………………………………23-24

Conclusion…………………………………………………………25

INTRODUCTION

Diabetes is the most common disease worldwide and keeps increasing everyday due to changing lifestyles, unhealthy food habits and overweight problems. Diabetes is a disease that occurs when your blood glucose, also called blood sugar, is too high. Using **K-NN** techniques to predict whether a Pima Indian Woman has diabetes or not, based on information about the patient such as blood pressure, body mass index (BMI), age, etc. Scientists carried out a study to investigate the significance of health-related predictors of diabetes in Pima Indian Women. The study population was females (21 years and above) of Pima Indian heritage. The purpose of the study was to find out the factors that are associated with the presence of diabetes in Pima Indians. To find out the reason behind this, we have to first analyze the relationship between different features, such as the number of times a woman was pregnant, their BMI, prevalence of diabetes, etc

DATA COLLECTION

The Data set used in this project is downloaded from **Kaggle website**. Which is a free source of Data sets formachine Learning and Data Science.
It is a reliable source , so we took data from Kaggle . The step of gathering data is the foundation ofthe machine learning process.

Dataset: preprocessed_diabetes_data.csv"

The dataset contains relevant There are total 769 rows and 9 columns(attributes) in the dataset.
Some of the attributes are Pregnancy's, Glucose, Blood Pressure, Skin Thickness , Insulin ,BMI, Age ,Outcome, Diabetic Pedigree Function.
After downloading it to the PC can be directly upload to the Google Colab.

```
files uploaded = files.upload( )
diabetes_data = pd.read_csv('preprocessed_diabetes_data.csv')
```

Libraries

We have imported few libraries which are needed for the whole process.

Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
```

1.NUMPY:- Numerical Python

import numpy as np

Numpy Python library is used for including any type of mathematical operation in the code. It is thefundamental package for scientific calculation in Python. It also supports to add large, multidimensional arrays and matrices. So, in Python, we can import it as:

2.SEABORN:-

import seaborn as sns

Seaborn is a library for **making statistical graphics** in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data

3.PANDAS:-

import pandas as pd

Pandas is **an open source library in Python**. It provides ready to use high-performance data structures and data analysis tools. Pandas module runs on top of NumPy and it is popularly used for data science and data analytics

4.MATPLOTLIB:-

import matplotlib.pyplot as plt

Matplotlib is a python library **used to create 2D graphs and plots by using python scripts**. It hasa module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc.

5.SCIKIT-LEARN:-

from sklearn.neighbours import KNeighborsClassifiers
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.

Exploratory Data Analysis(EDA)

After importing libraries and dataset we will do EDA. It is used to analyze the data using visual techniques. It is used   to discover trends, patterns to check assumptions with the   help of statistical summary and graphical representations.

1.View top 5 rows of dataset

```
diabetes_data.head(5)
```

	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

Identification of variables and data types

```
diabetes_data.shape
(768, 9)
```

Dataset comprises of 768 observations and 9 fields.

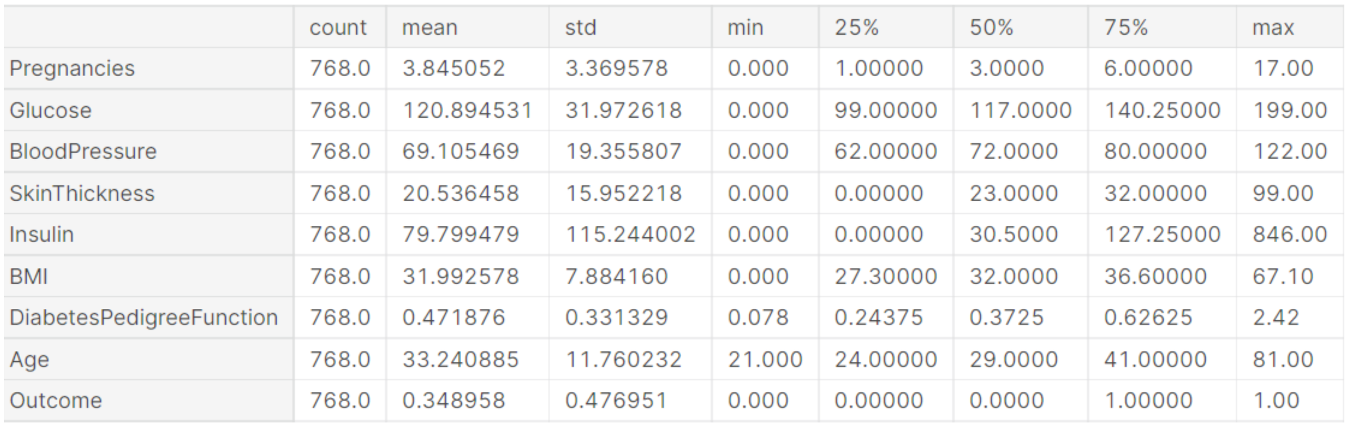
The following features have been provided to help us predict whether a person is diabetic or not:

- **Pregnancies:** Number of times pregnant
- **Glucose:** Plasma glucose concentration over 2 hours in an oral glucose tolerance test. Less than 140 mg/dL is considered normal level of glucose.

- **BloodPressure:** Diastolic blood pressure (mm Hg). 120/80 is normal BP level for female above 18 yr old.
- **SkinThickness:** Triceps skin fold thickness (mm)
- **Insulin:** 2-Hour serum insulin (mu U/ml). 16-166 mIU/L is considered the normal level of insulin.
- **BMI:** Body mass index (weight in kg/(height in m)2)
- **DiabetesPedigreeFunction:** Diabetes pedigree function (a function which scores likelihood of diabetes based on family history)
- **Age:** Age (years)
- **Outcome:** Class variable (0 if non-diabetic, 1 if diabetic)

Let's also make sure that our data is clean (has no null values, etc).

```
# Get the details of each column
diabetes_data.describe().T
```



	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

Missing Value

Missing data in the data set can reduce the power / fit of a model or can lead to a biased model because we have not analysed the behavior and relationship with other variables correctly. It can lead to wrong prediction or classification.

By analysing the above details of the dataset found that few features have zero values and pregnancy variable has maximum = 17 which seems to be impossible.

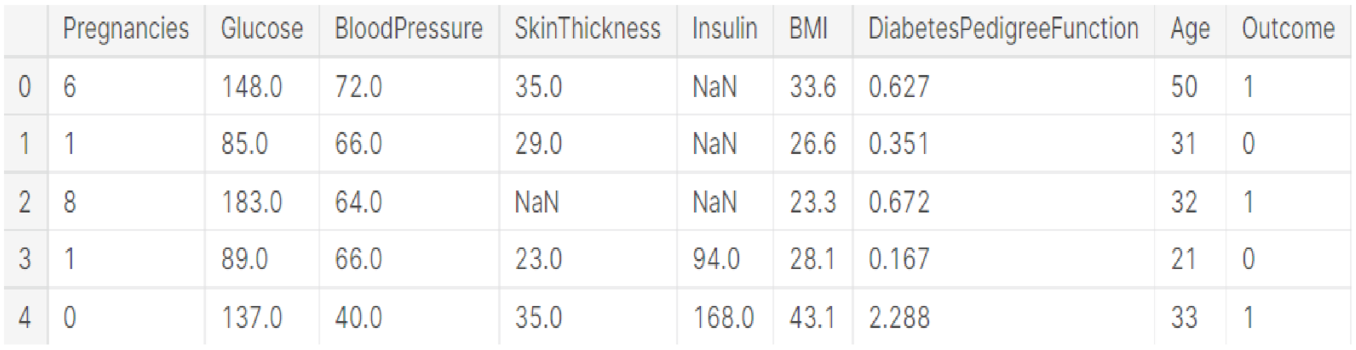
These column values of zero do not make sense as there is some range for a normal healthy human being which is certainly not the zero and thus indicates a missing value.

Below variables have an invalid zero value:

- Glucose
- BloodPressure
- SkinThickness
- Insulin
- BMI

Initially we will replace these zeros with NaN so that it will easy to count the missing values. Then, later on, we will replace them with appropriate values.

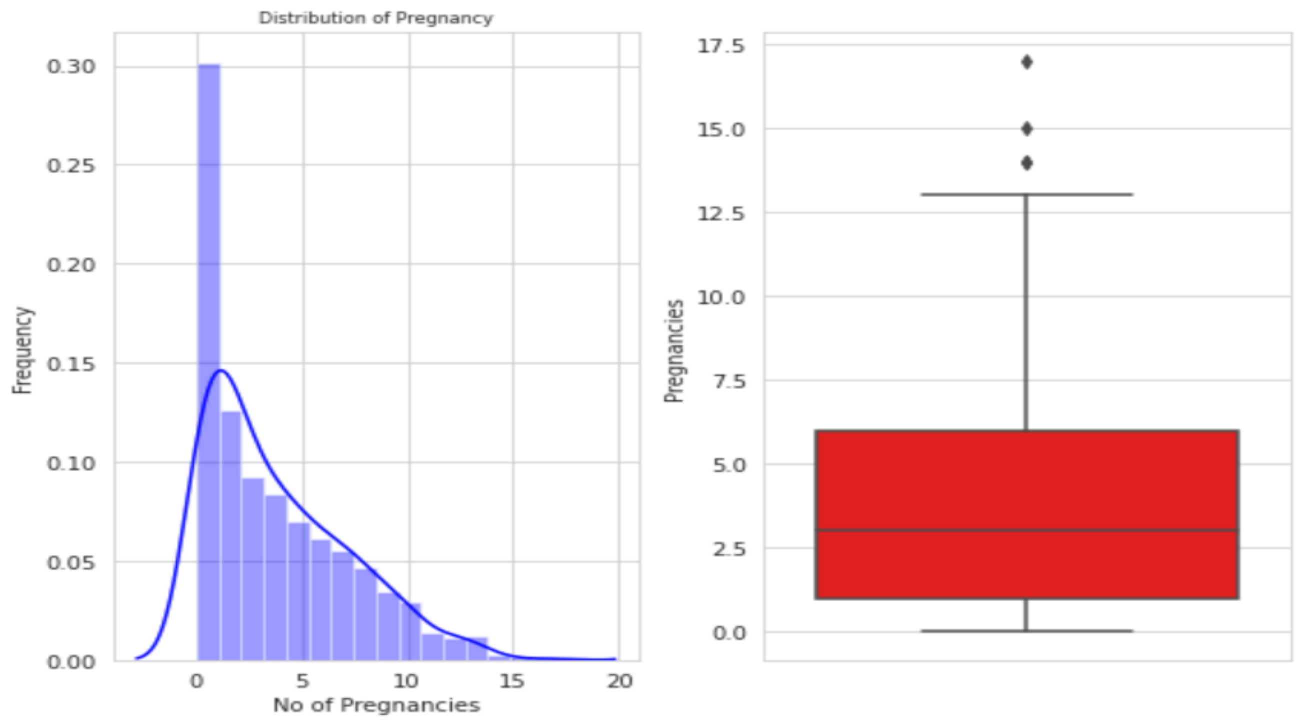
```
diabetes_data['Pregnancies'].value_counts()

# Replace zeros with NaN
diabetes_data[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = diabetes_data[['Gluc
ose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

Let us see distribution and also boxplot for outliers of feature "Pregnancies".

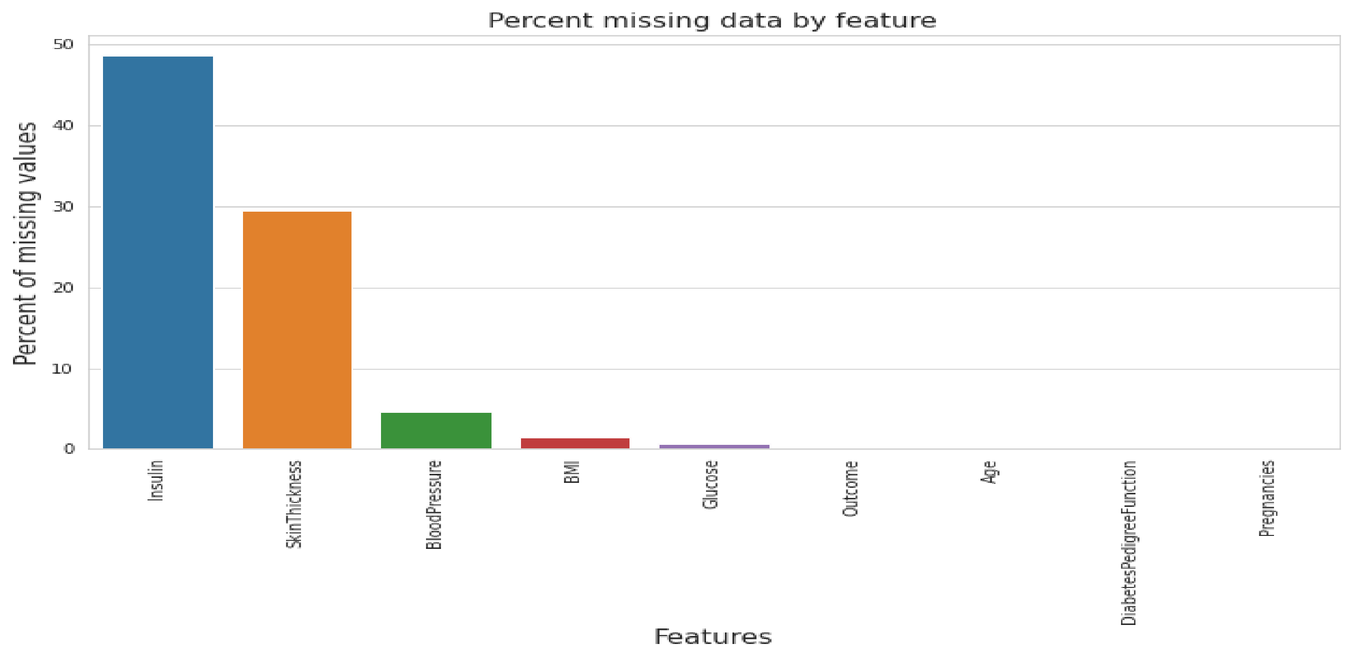
```
fig,axes = plt.subplots(nrows=1,ncols=2,figsize =
(8,6))zplot00=sns.distplot(diabetes_data['Pregnancies'],ax=axes[0],color='b')

axes[0].set_title('Distribution of Pregnancy',fontdict={'fontsize':8})
axes[0].set_xlabel('No of Pregnancies')
axes[0].set_ylabel('Frequency')
plt.tight_layout()


plot01=sns.boxplot('Pregnancies',data=diabetes_data,ax=axes[1],orient = 'v', color='r')
plt.tight_layout()
```



Missing values counts

```
total = diabetes_data.isnull().sum().sort_values(ascending=False)
percent = ((diabetes_data.isnull().sum()/diabetes_data.isnull().count())*100).sort_values(ascen
ding=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(9)
```

Percent missing data by feature



Highlights

- Insulin has 374 missing values which is about 48.7% of total missing values.
- SkinThickness has 227 missing values which is only 29.6% of total missing values.
- BloodPressure has 35 missing values which is only 4.6% of total missing values.
- BMI has only 11 missing values which is only 1.4% of total missing values

Data Distribution

Pearson's Correlation Coefficient

Pearson's Correlation Coefficient helps you find out the relationship between two quantities. It gives you the measure of the strength of association between two variables. The value of Pearson's Correlation Coefficient can be between -1 to +1. 1 means that they are highly correlated and 0 means no correlation.

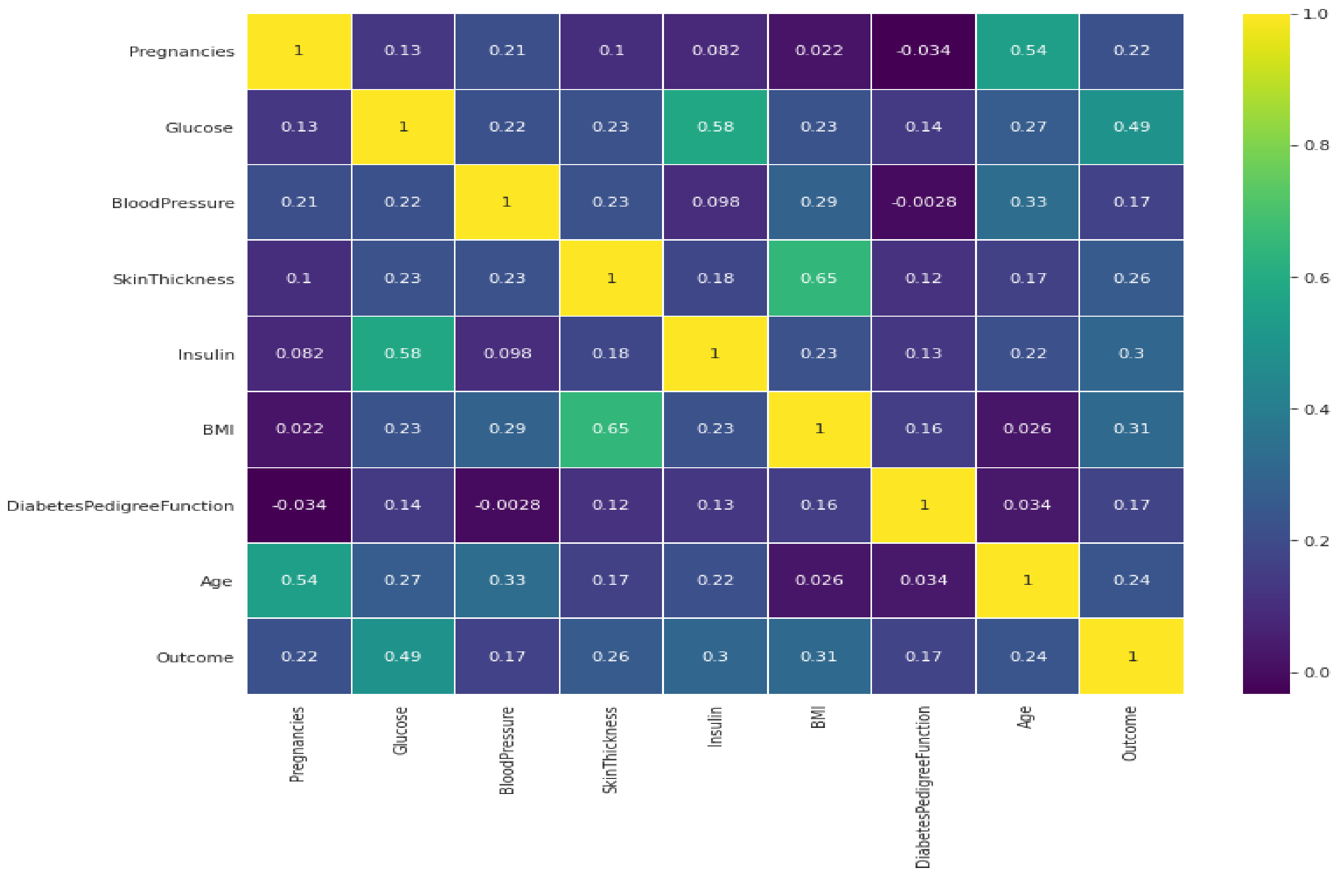
```
diabetes_data.corr()
```

Heatmap to find correlation

```
plt.figure(figsize=(12,10))
sns.heatmap(diabetes_data.corr(),annot=True, cmap='viridis',linewidths=.1)
plt.show()
```

Highlights

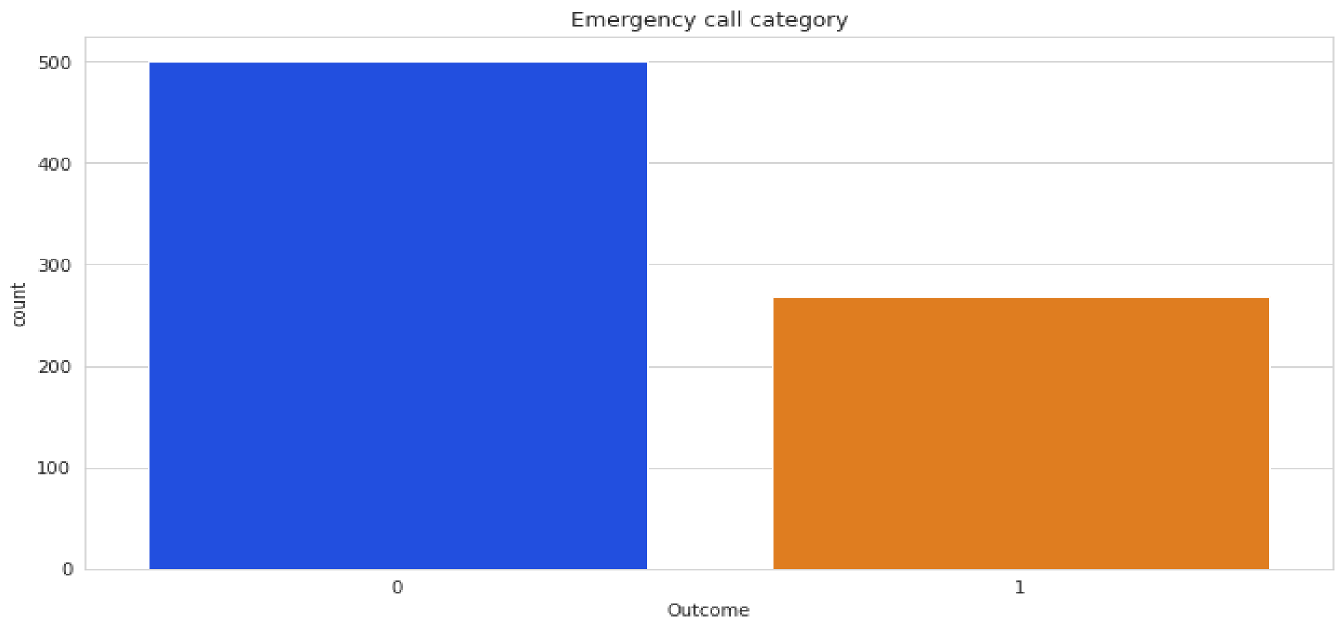
- It seems that Insulin is highly correlated with Glucose (about 0.58), BMI (about 0.23) and Age (about 0.22). It means that as the values of glucose, BMI and Age increase, the insuline is also increasing. It seems logical also that fat and aged people might have high level of insuline in their bodies.
- In the same way SkinThickness is highly correlated with BMI (about 0.65).



Checking balance of data

We can produce a seaborn count plot to see how the output is dominated by one of the classes or not.

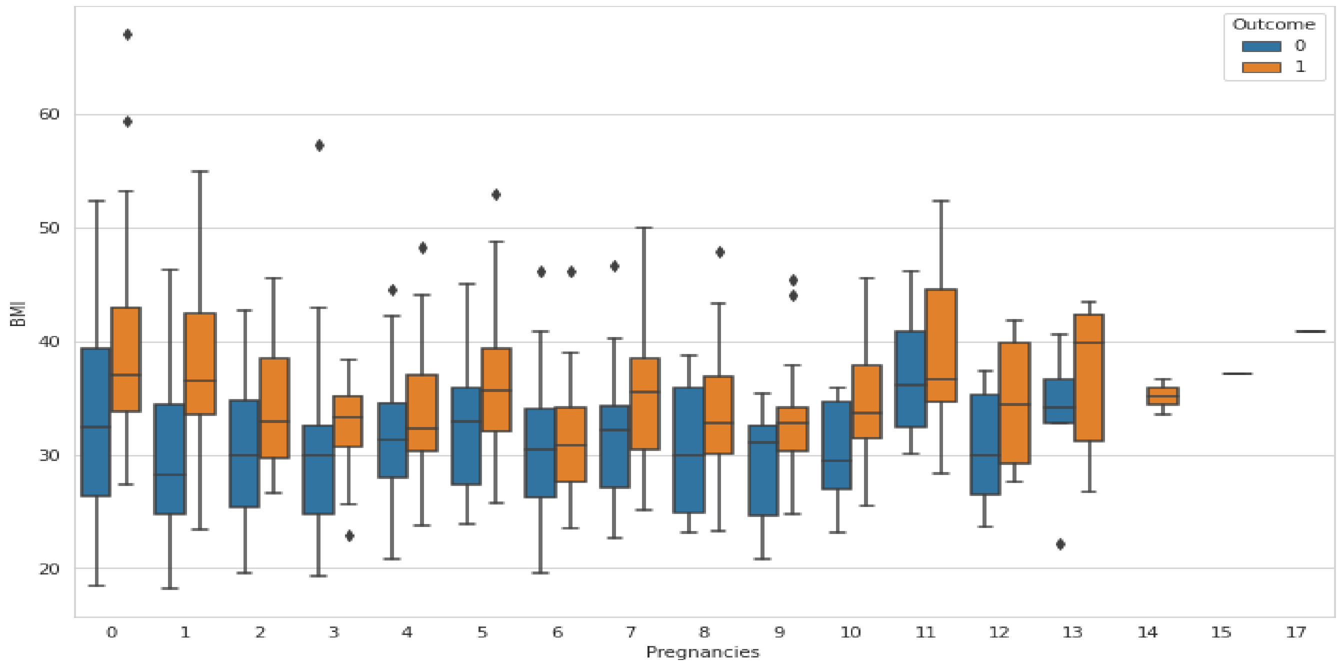
```
plt.figure(figsize=(12,6))
sns.countplot(x='Outcome',data=diabetes_data, palette='bright')
plt.title("Emergency call category")

print(diabetes_data['Outcome'].value_counts())
0    500
1    268
Name: Outcome, dtype: int64
```



Observation

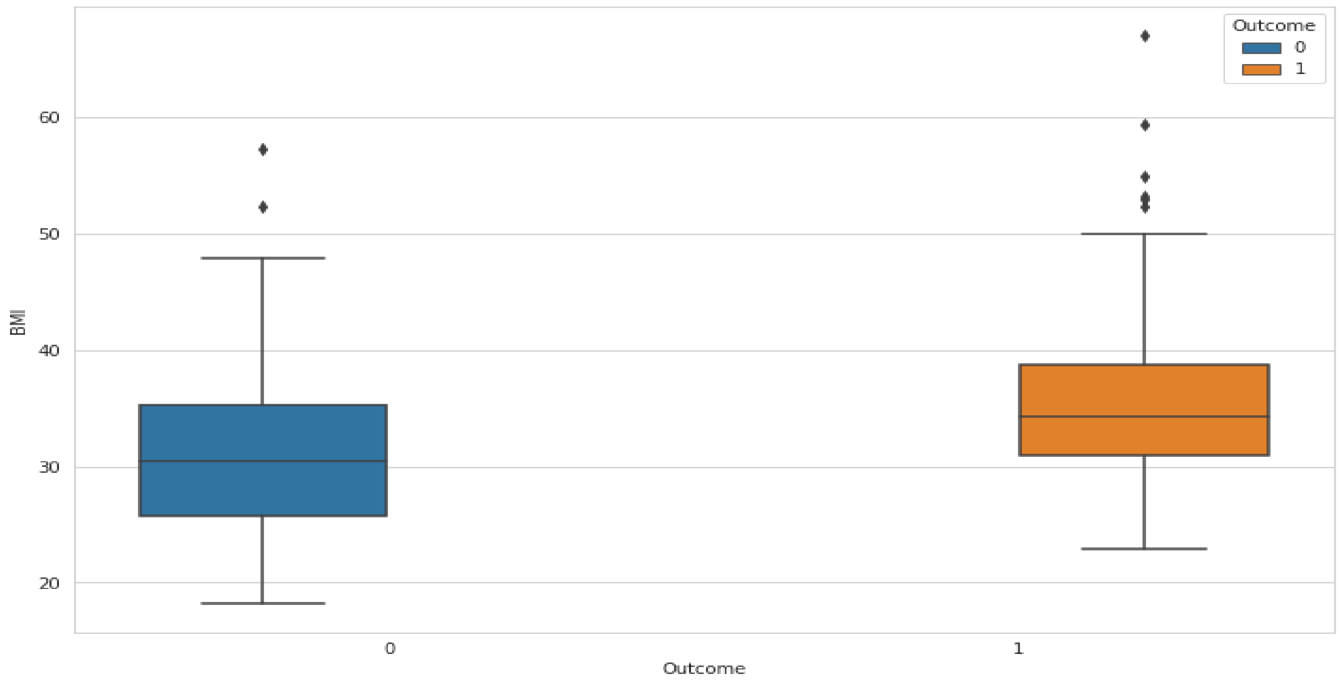
A total of 768 women were registered in the database. 268 womens about 35% were having diabetes, while 500 women about 65% were not.
The above graph shows that the dataset is biased towards non-diabetic patient. The number of non-diabetics is almost twice the number of diabetic patients.

```
plt.figure(figsize=(12,8))
sns.boxplot(x='Pregnancies', y='BMI',data=diabetes_data, hue='Outcome')
```



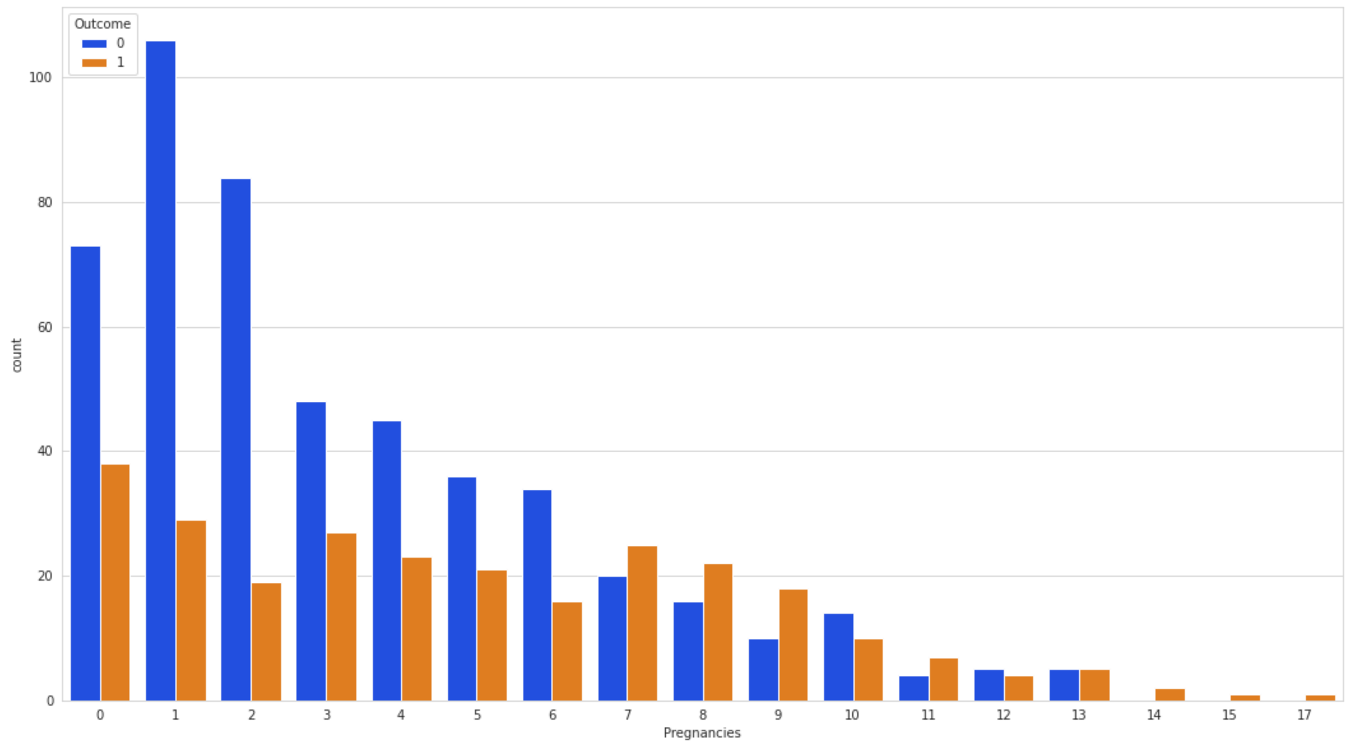
```
plt.figure(figsize=(12,8))
sns.boxplot(x='Outcome', y='BMI',data=diabetes_data, hue='Outcome')
```

Out[21]>



```
plt.figure(figsize=(18,10))
sns.countplot(x='Pregnancies',data=diabetes_data,hue = 'Outcome', palette='bright')
```

Out[22]:



Observations

The median BMI does not immensely change as the number of pregnancies increases.Those who tested positive for diabetes had higher BMIs than those who does not; yet, not a larger difference between the medians.

BMI will generally be higher for women who have had more numbers of pregnancy as well as for those who test positive for diabetes and that the relationship between the pedigree function and the test results will show that those who had a higher pedigree function tested positive and those who had a lower pedigree function tested negative.

Pregnancy vs Diabetes

```
plt.figure(figsize=(12,8))
sns.boxplot(x='Outcome', y='Pregnancies',data=diabetes_data)
```

Observations

The average number of pregnancies is higher (4.9) in diabetic in comparing to (3.3) in non-diabetic women with a significant difference between them.

BMI vs Diabetes

```
plt.figure(figsize=(12,8))
sns.boxplot(x='Outcome', y='BMI',data=diabetes_data)
```

Diabetic in Normal BMI

Let try to find out how is the probabiliy of having diabetic in a women having normal BMI. Please note that the range of noraml BMI is 18.5 to 25.

In [26]:

```
normalBMIData = diabetes_data[(diabetes_data['BMI'] >= 18.5) & (diabetes_data['BMI'] <= 25)]
normalBMIData['Outcome'].value_counts()
```

Out[26]:

```
0    101
1      7
Name: Outcome, dtype: int64
```

```
notNormalBMIData = diabetes_data[(diabetes_data['BMI'] < 18.5) | (diabetes_data['BMI'] > 25)]
notNormalBMIData['Outcome'].value_counts()
```

Out[27]:

```
0    399
1    261
Name: Outcome, dtype: int64
```

Observations

The Body Mass Index (BMI) showed a significant association with the occurrence of diabetes and that even the normal weighted women were at almost 9 times risk of being diabetic in comparison to the overweight.

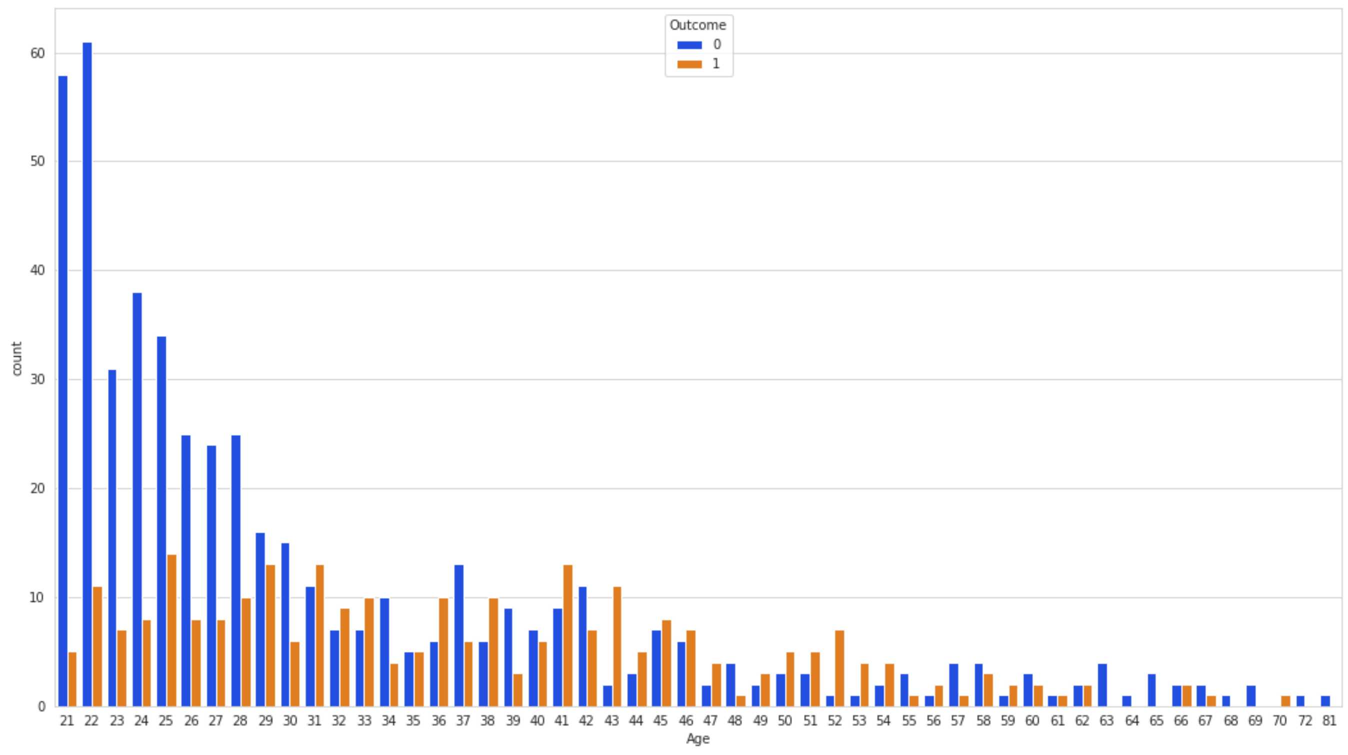
Age vs Diabetes

```
plt.figure(figsize=(12,8))
sns.boxplot(x='Outcome', y='Age',data=diabetes_data)
```

```
22    72
21    63
25    48
24    46
23    38
Name: Age, dtype: int64
```

In [31]:

```
plt.figure(figsize=(18,10))
sns.countplot(x='Age',data=diabetes_data,hue = 'Outcome', palette='bright')
```



Highlights

Significant relation can be seen between the age distribution and diabetic occurrence. Women at age group > 31 years were at higher risk to contract diabetes in comparison to the younger age group.

PREDICTION USING K-NN

KNN-classifier can be used when data set is small enough.

Standardize the Variables

Standardization (also called z-score normalization) is the process of putting different variables on the same scale. Standardization transforms your data such that the resulting distribution has a mean of 0 and a standard deviation of

$$Z = \frac{X - \mu}{\sigma}$$

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

In [33]:

```
scaler.fit(diabetes_data.drop('Outcome',axis=1))
```

Out[33]:

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [34]:

```
scaled_features = scaler.transform(diabetes_data.drop('Outcome',axis=1))
```

Convert the scaled features to a dataframe and check the head of this dataframe to make sure the scaling worked.

In [35]:

```
df_feat = pd.DataFrame(scaled_features,columns=diabetes_data.columns[:-1])
```

Train Test Split

Use train_test_split to split your data into a training set and a testing set.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(scaled_features,diabetes_data['Outcome'],test_size=0.30,random_state=101)
```

Create a KNN model instance with n_neighbors=1

```
knn = KNeighborsClassifier(n_neighbors=1)
```

Fit this KNN model to the training data.`

```
knn.fit(X_train,y_train)
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

Choosing a K Value

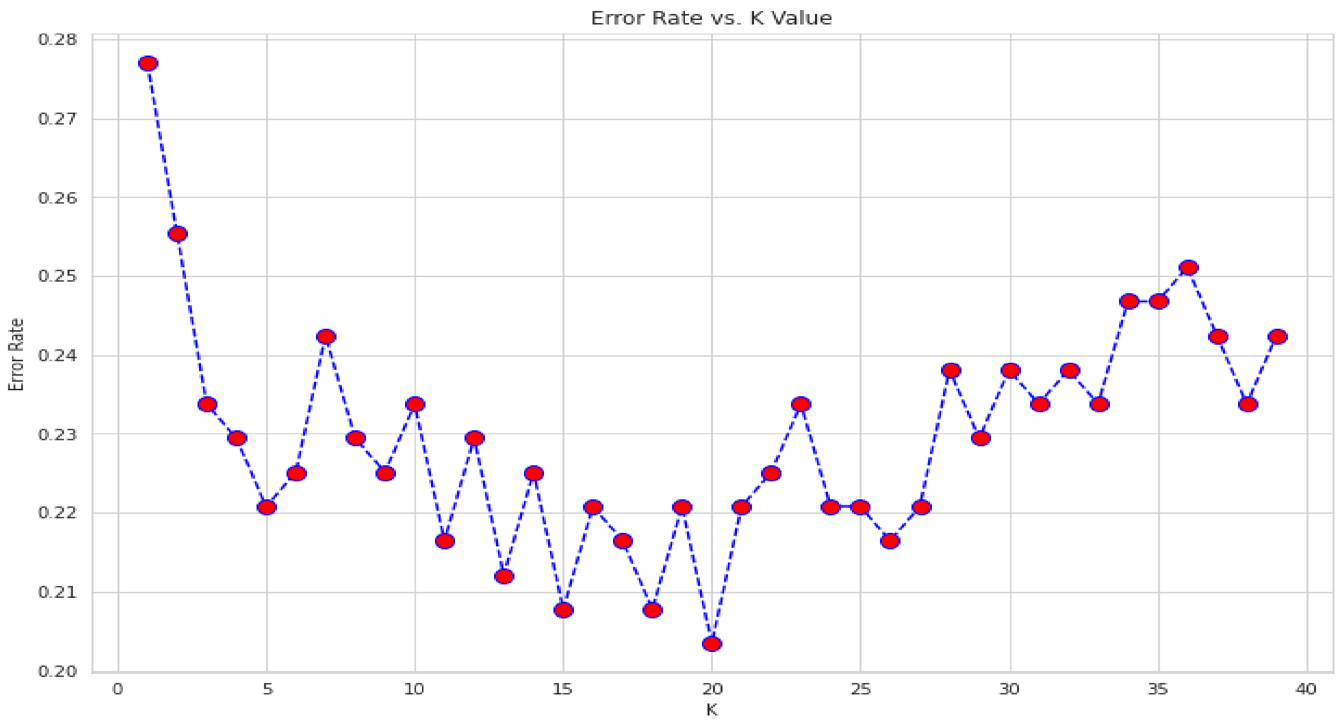
Let's go ahead and use the elbow method to pick a good K Value!

Create a for loop that trains various KNN models with different k values, then keep track of the error_rate for each of these models with a list. Refer to the lecture if you are confused on this step.

```
error_rate = []
test_scores = []
train_scores = []

for i in range(1,40):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
    train_scores.append(knn.score(X_train,y_train))
    test_scores.append(knn.score(X_test,y_test))
```

Now create the following plot using the information from your for loop.

```
plt.figure(figsize=(12,8))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
        markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
Text(0, 0.5, 'Error Rate')
```



```
max_train_score = max(train_scores)
train_scores_ind = [i for i, v in enumerate(train_scores) if v == max_train_score]
print('Max train score {} % and k = {}'.format(max_train_score*100,list(map(lambda x: x+1, train_scores_ind)
)))
Max train score 100.0 % and k = [1]

max_train_score = max(train_scores)
train_scores_ind = [i for i, v in enumerate(train_scores) if v == max_train_score]
print('Max train score {} % and k = {}'.format(max_train_score*100,list(map(lambda x: x+1, train_scores_ind)
)))

Max test score 79.65367965367966 % and k = [20]


Max test score 79.65367965367966 % and k = [20]
```

Retrain with new K Value

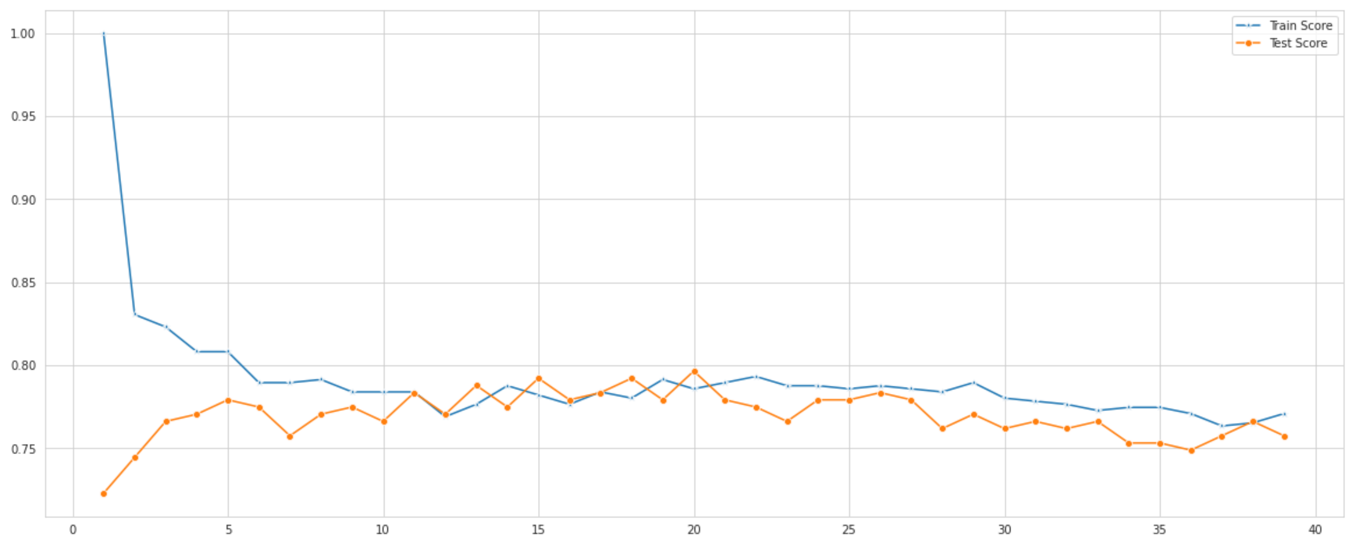
Retrain your model with the best K value and re-do the classification report and the confusion matrix.

```
# NOW WITH K=20
knn = KNeighborsClassifier(n_neighbors=20)

knn.fit(X_train,y_train)
pred = knn.predict(X_test)

print('WITH K=20')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

```
[[134  16]
 [ 31  50]]


              precision    recall  f1-score   support

           0       0.81      0.89      0.85       150
           1       0.76      0.62      0.68        81

    accuracy                           0.80       231
   macro avg       0.78      0.76      0.77       231
weighted avg       0.79      0.80      0.79       231
```

Result Visualisation

```
plt.figure(figsize=(20,8))
sns.lineplot(range(1,40),train_scores,marker='*',label='Train Score')
sns.lineplot(range(1,40),test_scores,marker='o',label='Test Score')
```



The best result is captured at k = 20 hence 20 is used for the final model

```
knn = KNeighborsClassifier(20)
```

```
knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

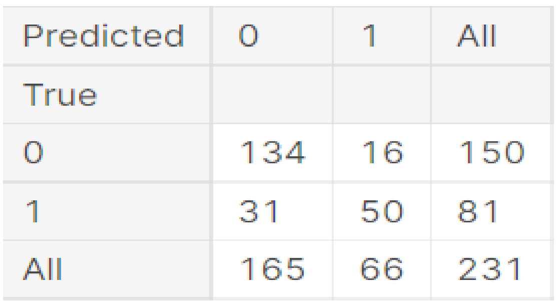
```
0.7965367965367965
```

Model Performance Analysis

Confusion Matrix

The confusion matrix is a technique used for summarizing the performance of a classification algorithm i.e. it has binary outputs.

```
#import confusion_matrix
from sklearn.metrics import confusion_matrix

y_pred = knn.predict(X_test)
confusion_matrix(y_test,y_pred)
pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
```



Predicted	0	1	All
True			
0	134	16	150
1	31	50	81
All	165	66	231

```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
p = sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="viridis" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 15.0, 'Predicted label')
```

Classification Report

Report which includes Precision, Recall and F1-Score.

Precision Score

Precision – Accuracy of positive predictions.

Recall Score

Recall(sensitivity or true positive rate): Fraction of positives that were correctly identified.

Recall=TP(TP+FN)

F1 Score

F1 Score – A helpful metric for comparing two classifiers. F1 Score takes into account precision and the recall. It is created by finding the the harmonic mean of precision and recall

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
Precision    recall  f1-score   support

           0       0.81      0.89      0.85       150
           1       0.76      0.62      0.68        81

    accuracy                           0.80       231
   macro avg       0.78      0.76      0.77       231
weighted avg       0.79      0.80      0.79       231
```

```
from sklearn import metrics

print("Accuracy of the model : {0:0.3f}".format(metrics.accuracy_score(y_test, y_pred)))
```

Accuracy of the model : 0.797

Conclusion

Overall, it seems that there is some form of an association between BMI, number of pregnancies, pedigree function, and the test results for diabetes.

It is surprising that the median BMI does not immensely change as the number of pregnancies increases. I expected there to be a strong positive relationship between the number of pregnancies and the BMI.

Those who tested positive for diabetes had higher BMIs than those who did not; yet, I predicted a larger difference between the medians.

As for the classification tasks, the standardized data yields much better results than the unscaled data over most of the K-values considered, thus indicating the importance of standardizing data in Machine Learning problems

Accuracy of the model is :0.797

Github link:

https://github.com/Dharani2905/miniproject/blob/380d9808a5007c4091c88c3b8dfc8dc9997927cf/mini_project.ipynb