BIS 634 DiabetesPrediction_Final Report_Yining Chen

Introduction

The objective of the dataset is to diagnostically predict whether a patient has diabetes, based on certain diagnostic measurements included in the dataset. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.

Diabetes is a serious health problem in the United States. According to the Centers for Disease Control and Prevention (CDC), more than 34 million people in the United States have diabetes, and about 90% of them have type 2 diabetes. This makes diabetes prediction a crucial point for providing a mature health care system. Diabetes prediction can help individuals identify their risk for developing diabetes and take steps to prevent or delay the onset of the disease. This can help individuals maintain their overall health and well-being, as well as reduce the burden of diabetes on the healthcare system.

What makes this dataset more interesting is that instead of focusing on a wide range of population, this data I found targets on a narrowed down demographic population. All patients here are females at least 21 years old of Pima Indian heritage. This project aims on studying the health condition on Pima Indian heritage and provide a prediction model for the indigenous people to improve their health care system.

Data Description

This dataset contains 9 variables, which is shown as below. The first 8 variables are independent variables which are used as predictors for diabetes. The outcome variable is a binary variable which indicates whether the patient has diabetes or not. When the outcome value is 1, it means that the patient has diabetes. When the outcome is 0, it means that the patient does not have diabetes. This dataset contains 768 patient records in total.

Columns	Description
Pregnancies	To express the Number of pregnancies
Glucose	To express the Glucose level in blood
BloodPressure	To express the Blood pressure measurement
SkinThickness	To express the thickness of the skin
Insulin	To express the Insulin level in blood
BMI	To express the Body mass index
DiabetesPedigreeFunction	To express the Diabetes percentage
Age	To express the age
Outcome	To express the final result 1 is Yes and 0 is No

Data Acquisition and FAIRness

I downloaded this dataset from Kaggle. This website is open sourced, which means that everyone is able to access and download it from this link: https://www.kaggle.com/datasets/whenamancodes/predict-diabities

In data processing, FAIRness is the abbreviation of Findable, Accessible, Interoperble, and Reusable. The diabetes dataset is from an opensource website, and everyone could download it using a Google account. In other words, the findability and accessibility are maintained. This dataset is stored in a csv file which makes it interoperbale because csv files are great for data processing and analyzing. The license of this dataset is listed as CC0, which is a public domain that allows the copyright holder to waive all rights to their work. This means that people can easily reuse this dataset and do multilple researches.

Data Cleaning and Preprocessing

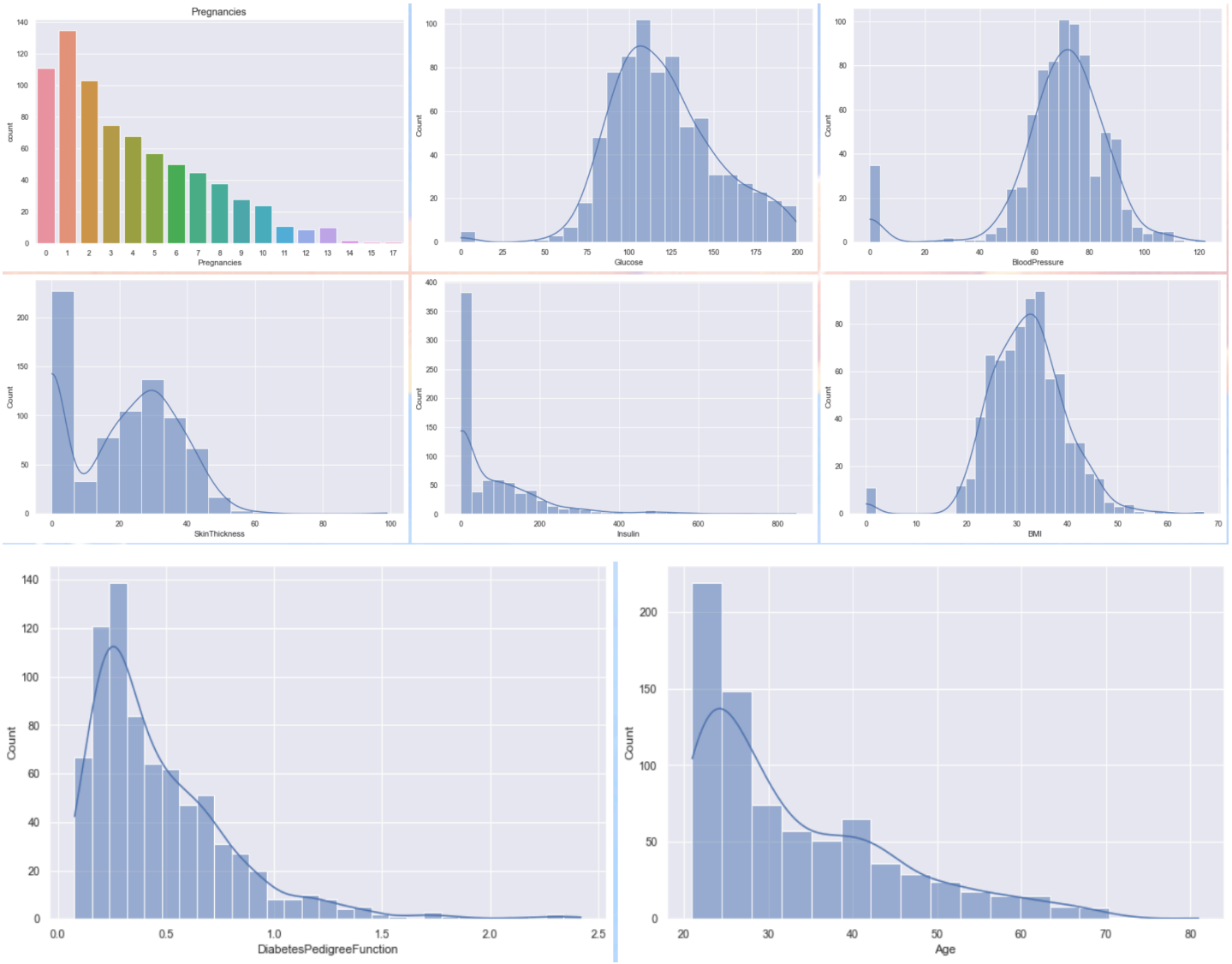
The orginal dataset is pretty cleaned without any NaN values. However, there are some suspicious values existed. As for a normal humanbeing, the blood pressure and skin thickness could never be 0. So I dropped all the rows with zero blood pressure or zero skin thickness. As for the insulin level, there are also some 0 values. However, previous researches addressed that it is possible for insulin level to be extremely low or even reach to zero. So we do not count these records with 0 insulin level as missing data and we still keep them in our data set. All the variables in this dataset are numerical values, which means that there's no need to handle dummy variables.

As for data preprocessing, we split the data in a ratio of 7:3 to set them as training data and testing data respectively. As for data standardization, we standardized the data when doing PCA analysis. But for our final analysis models, we did not standardize out data since after doing predition with both the original data and standardizing data, the standardization one did not provide better prediction.

Data Analysis

1. Issues with summary statistics

We plotted the histograms of each viriables to see the distribution of each columns (The plots are shown below).



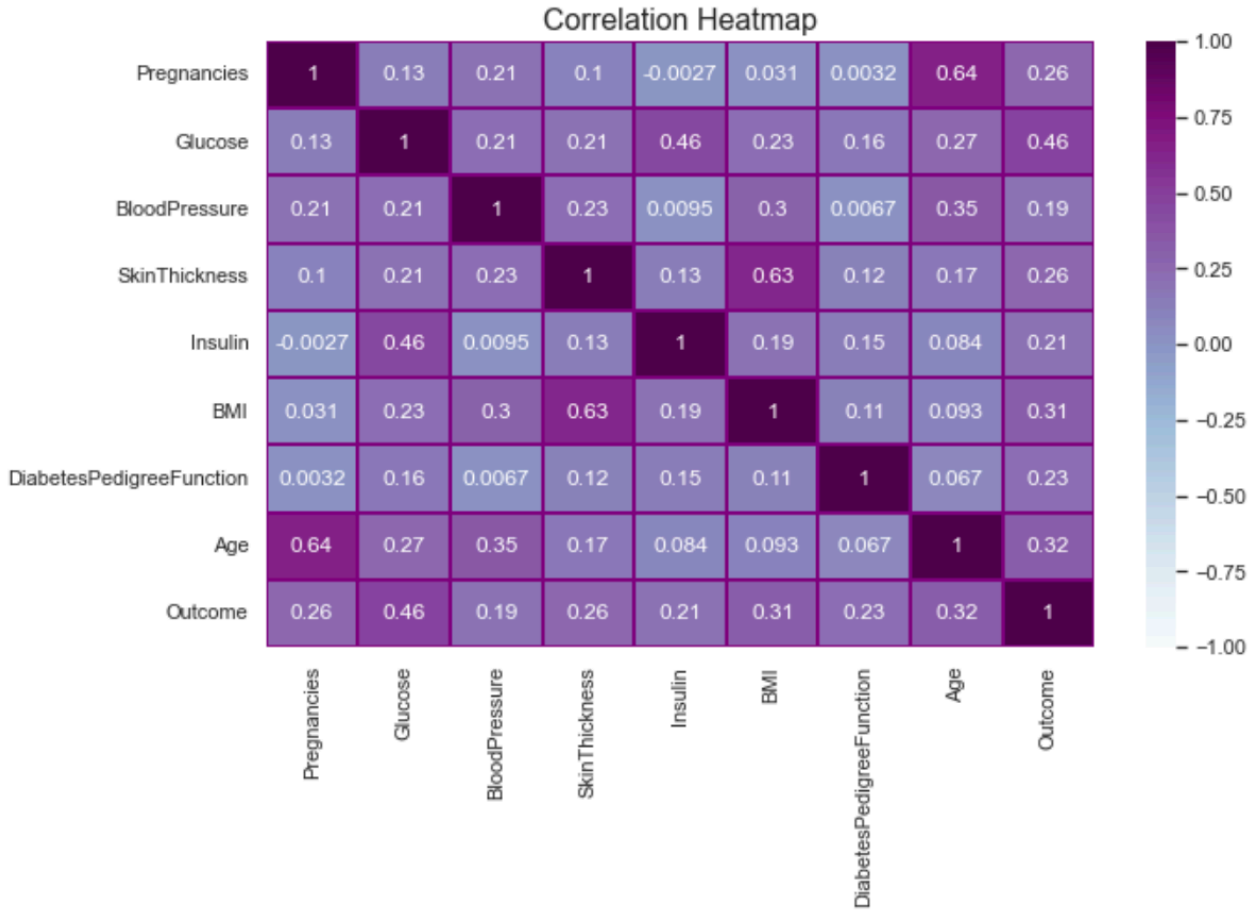
From the histograms above, we can conclude that most of our varibales are normally distributed (e.g.: Glucose, BloodPressure, SkinThickness, BMI). However, there are two variables that are extremly right skewed, which are Age and Pregnancies. We did not do any data tranformations to deal with the skewed distributions because it makes better sense that for this specific dataset, most pariticipants are young. And since the number of pregnancies is strongly correlated to the age of patient, it is reasonable that the numbe of pregnancy is skewed with a similar shape as Age.

2. Analysis

We did several analysis for this project including: correlation analysis, PCA analysis, within sample prediction, and out of sample prediction. As for within sample prediction, we implemented for different machine learning algorithms: logistic regression classification, random forest classification, XGBoost classification, and K-Nearest Neighbors classification. We chose to run these analysis because the Outcome variable is binary, which is perfect for machine learning classification. For each machine learning model, we printed the classification report and confusion matrix to do analysis validation.

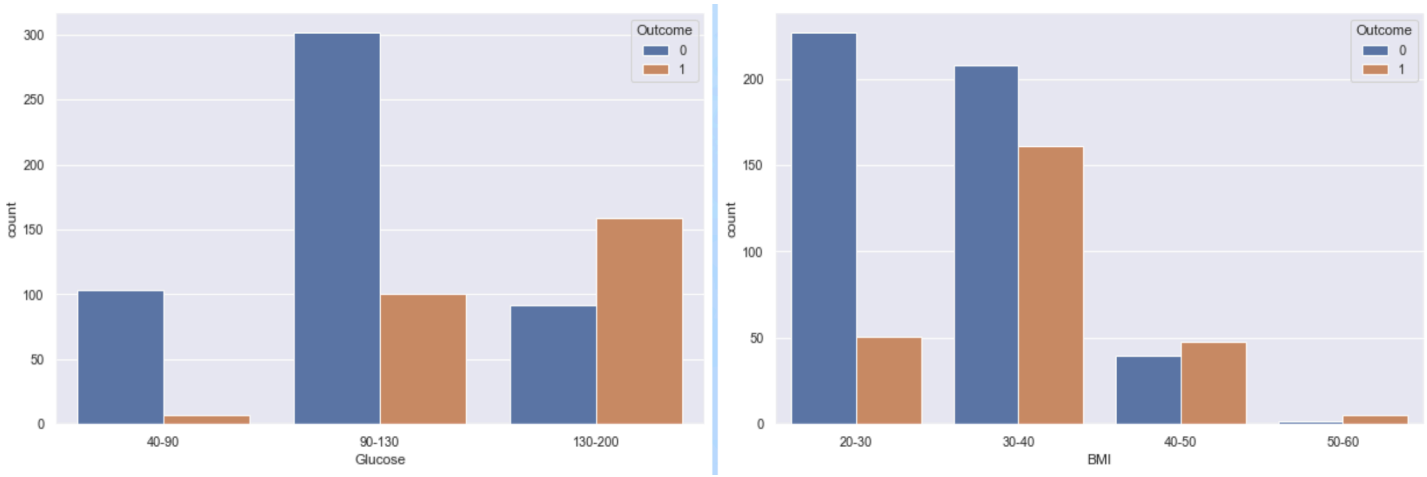
2.1 Correlation Analysis

We plotted a heatmap to represent the correlation between each variables. Then we listed all the correlation with Outcome in a descending order.




```
Outcome                    1.000000
Glucose                    0.464195
Age                        0.321420
BMI                        0.305894
SkinThickness              0.259199
Pregnancies                0.258883
DiabetesPedigreeFunction   0.226655
Insulin                    0.209954
BloodPressure              0.185618
```

From the graphs above, we could see that glucose level, age, and BMI are more correlated with the outcome of diabetes. However, only using the age to predict diabetes is not a reasonable method. So we plotted count plots to see how the glucose level and BMI index could influence the outcome of diabetes. The graphs are show below.



From the graphs above, we could conclude that as for glucose level, we can see that (40, 130) could be considered as a relatively safe range. When the patient's glucose level exceeds 130, there's a higher risk that this patient has diabetes. When the BMI index of the patient is the the range of (20, 30), there's a relatively low risk of having diabetes. When the BMI index is in the range of (40, 50), more than half of patients are diagnosed with diabetes. When the BMI index exceeds 50, the patient is most likely having diabetes.

2.2 Logistic Regression Classification, Random Forest Classification, and XG Boost Classification

To validate our analysis, we implemented confusion matrix and classification report to see the accuracy of our models. The results are shown as below.

Logistic Regression Training Data Result:

Confusion Matrix

	Normal	Diabetes
Normal	262	25
Diabetes	64	80

Classification Report

	0	1	macro avg	micro avg	weighted avg
precision	0.804	0.762	0.794	0.783	0.79
recall	0.913	0.556	0.794	0.734	0.794
f1-score	0.855	0.643	0.794	0.749	0.784
support	287.0	144.0	0.794	431.0	431.0

Logistic Regression Testing Data Result:

Confusion Matrix

	Normal	Diabetes
Normal	64	9
Diabetes	20	15

Classification Report

	0	1	macro avg	micro avg	weighted avg
precision	0.762	0.625	0.731	0.693	0.718
recall	0.877	0.429	0.731	0.653	0.731
f1-score	0.815	0.508	0.731	0.662	0.716
support	73.0	35.0	0.731	108.0	108.0

As for training data, There are **262** normal people correctly predicted as normal and there are **25** noramal people incorrectly labeled as diabetes. There are **64** diabetes incorrectly labeled as normal, and there are **80** diabetes correctly labeled as diabetes.

As for testing data, There are **64** normal people correctly predicted as normal and there are **9** noramal people incorrectly labeled as diabetes. There are **20** diabetes incorrectly labeled as normal, and there are **15** diabetes correctly labeled as diabetes.

As for our training data, the accuracy of the prediction model is **0.794**. As for our testing data, the accuracy of the prediction model is **0.731**.

Random Forest Training Data Result:

Confusion Matrix

	Normal	Diabetes
Normal	287	0
Diabetes	0	144

Classification Report

	0	1	macro avg	micro avg	weighted avg
precision	1.0	1.0	1.0	1.0	1.0
recall	1.0	1.0	1.0	1.0	1.0
f1-score	1.0	1.0	1.0	1.0	1.0
support	287.0	144.0	1.0	431.0	431.0

Random Forest Testing Data Result:

Confusion Matrix

	Normal	Diabetes
Normal	57	16
Diabetes	13	22

Classification Report

	0	1	macro avg	micro avg	weighted avg
precision	0.814	0.579	0.731	0.697	0.738
recall	0.781	0.629	0.731	0.705	0.731
f1-score	0.797	0.603	0.731	0.7	0.734
support	73.0	35.0	0.731	108.0	108.0

As for training data, There are **287** normal people correctly predicted as normal and there are **0** noramal people incorrectly labeled as diabetes. There are **0** diabetes incorrectly labeled as normal, and there are **144** diabetes correctly labeled as diabetes.

As for testing data, There are **57** normal people correctly predicted as normal and there are **16** noramal people incorrectly labeled as diabetes. There are **13** diabetes incorrectly labeled as normal, and there are **22** diabetes correctly labeled as diabetes.

As for our training data, the accuracy of the prediction model is **1.0**. As for our testing data, the accuracy of the prediction model is **0.731**.

XGBoost Training Data Result:

Confusion Matrix

	Normal	Diabetes
Normal	287	0
Diabetes	0	144

Classification Report

	0	1	macro avg	micro avg	weighted avg
precision	1.0	1.0	1.0	1.0	1.0
recall	1.0	1.0	1.0	1.0	1.0
f1-score	1.0	1.0	1.0	1.0	1.0
support	287.0	144.0	1.0	431.0	431.0

XGBoost Testing Data Result:

Confusion Matrix

	Normal	Diabetes
Normal	55	18
Diabetes	16	19

Classification Report

	0	1	macro avg	micro avg	weighted avg
precision	0.775	0.514	0.685	0.644	0.69
recall	0.753	0.543	0.685	0.648	0.685
f1-score	0.764	0.528	0.685	0.646	0.687
support	73.0	35.0	0.685	108.0	108.0

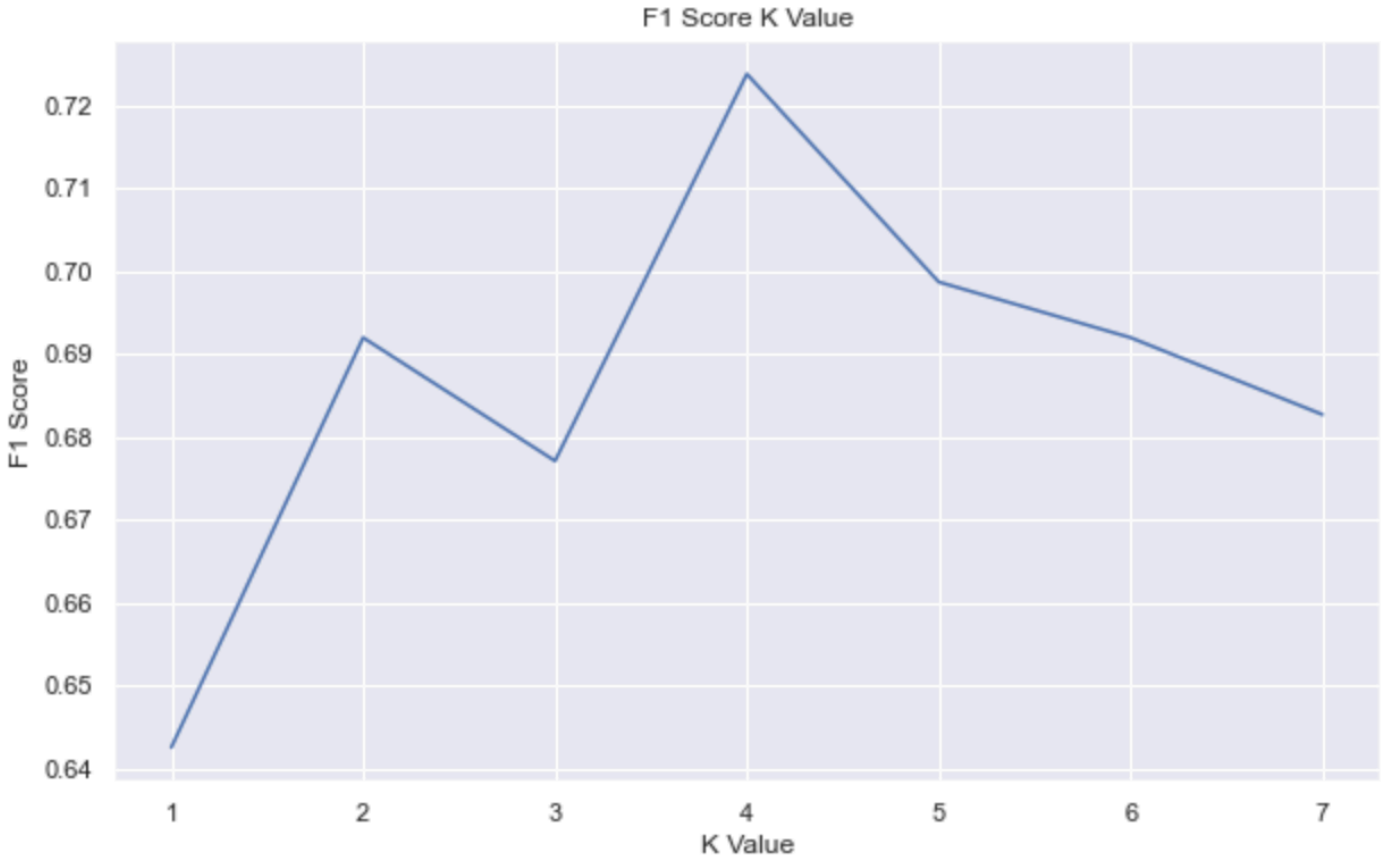
As for training data, There are **287** normal people correctly predicted as normal and there are **0** noramal people incorrectly labeled as diabetes. There are **0** diabetes incorrectly labeled as normal, and there are **144** diabetes correctly labeled as diabetes.

As for testing data, There are **55** normal people correctly predicted as normal and there are **18** noramal people incorrectly labeled as diabetes. There are **16** diabetes incorrectly labeled as normal, and there are **19** diabetes correctly labeled as diabetes.

As for our training data, the accuracy of the prediction model is **1.0**. As for our testing data, the accuracy of the prediction model is **0.685**.

2.3 K-Nearest Neighbors Classification

We did a line plot to see how the value of **k** could influence the accuracy of our KNN prediction models. By setting f1-score as the criteria, we generated the plot as shown below. The maximum accuracy evaluating by f1-score is **0.7237** and the maximum occurs when **k = 4**



Then, we did cross validation on k = 4. The classifaction report and confusion matrix is shown as below.

K-Nearest Neighbors Training Data Result:

Confusion Matrix

	Normal	Diabetes
Normal	274	13
Diabetes	65	79

Classification Report

	0	1	macro avg	micro avg	weighted avg
precision	0.808	0.859	0.819	0.833	0.825
recall	0.955	0.549	0.819	0.752	0.819
f1-score	0.875	0.669	0.819	0.772	0.807
support	287.0	144.0	0.819	431.0	431.0

K-Nearest Neighbors Testing Data Result:

Confusion Matrix

	Normal	Diabetes
Normal	65	8
Diabetes	20	15

Classification Report

	0	1	macro avg	micro avg	weighted avg
precision	0.765	0.652	0.741	0.708	0.728
recall	0.89	0.429	0.741	0.659	0.741
f1-score	0.823	0.517	0.741	0.67	0.724
support	73.0	35.0	0.741	108.0	108.0

As for training data, There are **274** normal people correctly predicted as normal and there are **13** noramal people incorrectly labeled as diabetes. There are **65** diabetes incorrectly labeled as normal, and there are **79** diabetes correctly labeled as diabetes.

As for testing data, There are **65** normal people correctly predicted as normal and there are **8** noramal people incorrectly labeled as diabetes. There are **20** diabetes incorrectly labeled as normal, and there are **15** diabetes correctly labeled as diabetes.

As for our training data, the accuracy of the prediction model is **0.819**. As for our testing data, the accuracy of the prediction model is **0.741**.

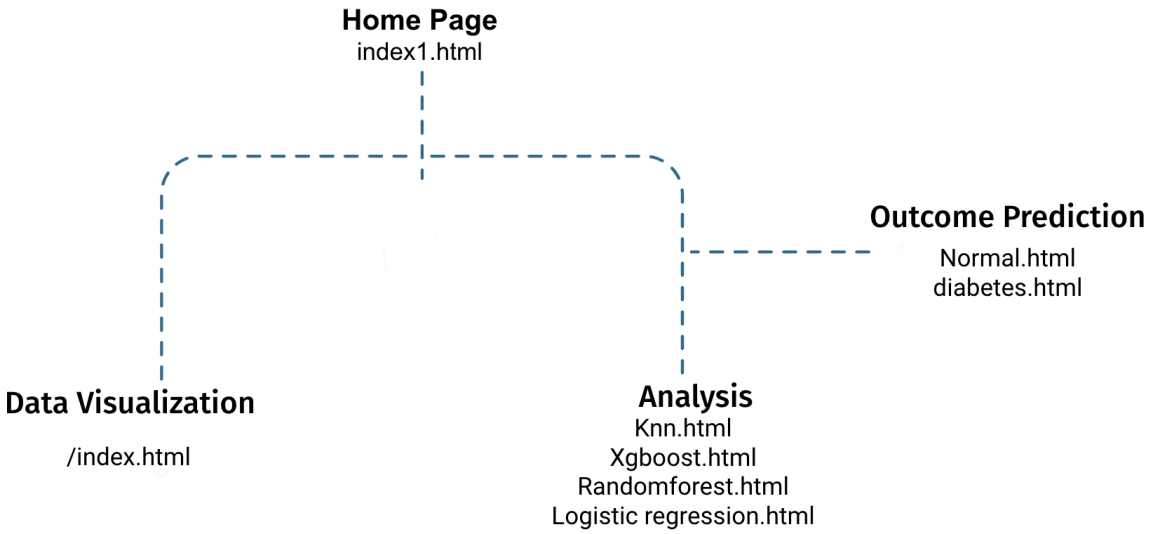
2.4 Results based on the classification models

By running the above models multiple times, based on the accuracy level on testing dataset, we could conclude that logistic regression classification could give us the best accuracy level in general. It is worth mentioning that the KNN classification and random forest classification models could also provide a relativly high level of accuracy.

One suprising finding is that all of our models are good at classifying normal people (with an accuracy level greater than 80%). However, if the participant has diabetes, there's a relatively higher rate of false negative. This could be counted as the biggest limitation of our models.

Web API

We used Flask combined with HTML to develop our the webpage. The following graph is the structure of the web API.



Three main functions of this website is to provide data visualization, data analysis (data training and within sample prediction), and outcome prediction (out of sample prediction).

The home page is written in index1.html file. It provides the dataset description, the source of dataset, variable dictionary, and buttons to direct into the three main functions.

The data visualization page is written in index.html file. It provides the dataset description, the source of data, variables dictionary, and the data visualization outputs. The data visualization output includes the histogram of each variables, the heatmap and the correlation table, the count plot, and the pca plot. At the bottom of this page, there's a "Return Home" button which could direct back to the home page.

The analysis part contains four different html files, including knn.html, xgboost.html, randomforest, and logisticregression.html. When the user specifies the model name by selecting the option bar in the homepage, the website will direct the user into different corresponding analysis pages. In different analysis page, we provide the basic description of each machine learning model, a button which linked to a outside webpage where the users could learn how to implement the machine learning algorithm themselves, and the cross validation results (including confusion matrix, classification report, and several interpretations). At the buttom of each page, there's a "Return Home" button which could direct back to the home page.

It is worth mentioning that we create the user interactive function by letting the user to choose the value of **k** for KNN classification analysis. By passing the value of k as the parameter, the user could easily see different analysis output for different values of k. We set the recommended k value as 4 because as we previously addressed, k = 4 provides the best prediction accuracy for this dataset.

As for the out of sample prediction (outcome prediction function), we provide textboxes for users to input their health data and use our trained logistic regression model model to check whether the user is diabetes or not. If the user is classified as normal, the webpage would direct the user to the normal.html file. Similarly, if the user is classified as diabetes, the webpage will direct the user to diabetes.html. Both normal and diabetes files contains a "Return Home" button which could direct back to the home page.

The python code of the server API is shown below:

```
from flask import Flask, render_template, request
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score, KFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb

df = pd.read_csv("diabetes.csv")
columns = ['BloodPressure', 'SkinThickness', 'DiabetesPedigreeFunction']
df = df.replace(0, pd.np.nan).dropna(subset=columns).fillna(0)

quant_cols_1 = df.drop('Outcome', axis=1)
Outcome = df['Outcome']

x_cols = df.columns[:(len(df.columns)-1)]
x_train, x_test, y_train, y_test = train_test_split(quant_cols_1, Outcome, test_size=0.3,
random_state = 42)
#scaler=StandardScaler()
X=df.drop(columns=["Outcome"])
#X=scaler.fit_transform(X)
y = df["Outcome"]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state = 42)


app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index1.html")
```

```
@app.route("/data", methods=["GET"])
def data():
    return render_template("index.html")

@app.route("/predict", methods=["GET"])
def predict():
    a = request.args.get("preg")
    b = request.args.get("glu")
    c = request.args.get("bp")
    d = request.args.get("st")
    e = request.args.get("i")
    f = request.args.get("bmi")
    g = request.args.get("dpf")
    h = request.args.get("age")
    X_test1 = [[a,b,c,d,e,f,g,h]]
    #X_test1 = [[6,148,72,35,0,33.6,0.627,50]]
    #X_test2 = [[2,141,58,34,128,25.4,0.699,24]]
    logistic_reg = LogisticRegression(solver='liblinear')
    logistic_reg.fit(X_train, y_train)
    pred = logistic_reg.predict(X_test1)
    if pred[0] == 1.0:
        return render_template("prediction2.html")
    elif pred[0] == 0.0:
        return render_template("prediction.html")
    return
@app.route("/info", methods=["GET"])
def info():
    usertext = request.args.get("models")
    if usertext == "K-Nearest Neighbors Prediction Model":
        k_val = int(request.args.get("K-Nearest"))
        knn = KNeighborsClassifier(k_val)
        knn.fit(X_train, y_train)
        prediction = knn.predict(X_test)
        report = pd.DataFrame(classification_report(y_test, prediction, output_dict=True))
        report = report.round(3)

        m = confusion_matrix(y_test, prediction)
        ul=m[0][0]
        ur=m[0][1]
        ll=m[1][0]
        lr=m[1][1]
        accuracy_test = round((ul+lr)/(ul+lr+ll+ur), 3)

        prediction_trained = knn.predict(X_train)
```

```
        report_train = pd.DataFrame(classification_report(y_train, prediction_trained,
output_dict=True))
        report_train = report_train.round(3)
        m_1 = confusion_matrix(y_train, prediction_trained)
        ul_1=m_1[0][0]
        ur_1=m_1[0][1]
        ll_1=m_1[1][0]
        lr_1=m_1[1][1]
        accuracy_trained = round((ul_1+lr_1)/(ul_1+lr_1+ll_1+ur_1), 3)
        return
render_template("knn.html",k_val=k_val,report_train=report_train,report=report,
accuracy_test = accuracy_test, accuracy_trained = accuracy_trained, ul=ul, ur=ur,ll=ll,
lr=lr, ul_1=ul_1, ur_1=ur_1,ll_1=ll_1, lr_1=lr_1)

    elif usertext == "Random Forest Prediction Model":
        random_forest = RandomForestClassifier(n_estimators=200)
        random_forest.fit(X_train, y_train)
        prediction = random_forest.predict(X_test)
        report = pd.DataFrame(classification_report(y_test, prediction, output_dict=True))
        report = report.round(3)

        m = confusion_matrix(y_test, prediction)
        ul=m[0][0]
        ur=m[0][1]
        ll=m[1][0]
        lr=m[1][1]
        accuracy_test = round((ul+lr)/(ul+lr+ll+ur), 3)

        prediction_trained = random_forest.predict(X_train)
        report_train = pd.DataFrame(classification_report(y_train, prediction_trained,
output_dict=True))
        report_train = report_train.round(3)
        m_1 = confusion_matrix(y_train, prediction_trained)
        ul_1=m_1[0][0]
        ur_1=m_1[0][1]
        ll_1=m_1[1][0]
        lr_1=m_1[1][1]
        accuracy_trained = round((ul_1+lr_1)/(ul_1+lr_1+ll_1+ur_1), 3)
        return
render_template("random_forest.html",report_train=report_train,report=report,
accuracy_test = accuracy_test, accuracy_trained = accuracy_trained, ul=ul, ur=ur,ll=ll,
lr=lr, ul_1=ul_1, ur_1=ur_1,ll_1=ll_1, lr_1=lr_1)

    elif usertext == "Logistic Regression Prediction Model":
        logistic_reg = LogisticRegression(solver='liblinear')
        logistic_reg.fit(X_train, y_train)
```

```
        prediction = logistic_reg.predict(X_test)
        report = pd.DataFrame(classification_report(y_test, prediction, output_dict=True))
        report = report.round(3)

        m = confusion_matrix(y_test, prediction)
        ul=m[0][0]
        ur=m[0][1]
        ll=m[1][0]
        lr=m[1][1]
        accuracy_test = round((ul+lr)/(ul+lr+ll+ur), 3)

        prediction_trained = logistic_reg.predict(X_train)
        report_train = pd.DataFrame(classification_report(y_train, prediction_trained,
output_dict=True))
        report_train = report_train.round(3)
        m_1 = confusion_matrix(y_train, prediction_trained)
        ul_1=m_1[0][0]
        ur_1=m_1[0][1]
        ll_1=m_1[1][0]
        lr_1=m_1[1][1]
        accuracy_trained = round((ul_1+lr_1)/(ul_1+lr_1+ll_1+ur_1), 3)
        return
render_template("logistic_reg.html",report_train=report_train,report=report, accuracy_test
= accuracy_test, accuracy_trained = accuracy_trained, ul=ul, ur=ur,ll=ll, lr=lr,
ul_1=ul_1, ur_1=ur_1,ll_1=ll_1, lr_1=lr_1)

    elif usertext == "XGBoost Prediction Model":
        xgb_model = xgb.XGBClassifier(objective="binary:logistic", random_state=42)
        xgb_model.fit(X_train, y_train)
        prediction = xgb_model.predict(X_test)
        report = pd.DataFrame(classification_report(y_test, prediction, output_dict=True))
        report = report.round(3)

        m = confusion_matrix(y_test, prediction)
        ul=m[0][0]
        ur=m[0][1]
        ll=m[1][0]
        lr=m[1][1]
        accuracy_test = round((ul+lr)/(ul+lr+ll+ur), 3)

        prediction_trained = xgb_model.predict(X_train)
        report_train = pd.DataFrame(classification_report(y_train, prediction_trained,
output_dict=True))
        report_train = report_train.round(3)
        m_1 = confusion_matrix(y_train, prediction_trained)
        ul_1=m_1[0][0]
```

```
        ur_1=m_1[0][1]
        ll_1=m_1[1][0]
        lr_1=m_1[1][1]
        accuracy_trained = round((ul_1+lr_1)/(ul_1+lr_1+ll_1+ur_1), 3)
        return render_template("xgboost.html",report_train=report_train,report=report,
accuracy_test = accuracy_test, accuracy_trained = accuracy_trained, ul=ul, ur=ur,ll=ll,
lr=lr, ul_1=ul_1, ur_1=ur_1,ll_1=ll_1, lr_1=lr_1)
    return


if __name__ == "__main__":
    app.run(debug=True)
```

Discussion and Limitation

There are serveral limitations of our study. One limitation is that the PCA analysis does not provide a reasonable classification for our dataset. The potential reason is that the correlation between each independent variables and the outcome variable is not large enough.

Another limitation of this study which might affect the accuracy of the models is that the dataset is not big enough. The patient record of the original dataset is 768 rows, and after data cleaning, there are only 539 entries. For the further research, adding more patient data would increase the accuracy level of our model. As we previously mentioned, there is a relatively higher false negative rate for our training datasets. Our further work would focus on minimizing the false negative rate.

Overall, the accuracy level of all our models are greater than 75%, which could be considered as a reasonable prediction precision level. We believe that these prediction models and the webpage could provide a better understanding on Indian Pima Female health conditions, which is useful for further research study and providing better health care for the indigenous people.