CPE 4040 Final: Data Analysis Project

In this semi-guided project, you will apply the skills that you learned in this semester to analyze a real-world dataset.

There are four parts in this assignment:

1. Data preparation and cleaning
2. Exploratory data analysis and visualization
3. In-depth analysis
4. Regression model creation and outcome prediciton

General guidelines:

- Do each part of the project in a clean and logical manner.
- Make comments on your codes. Make insightful observations after the analysis.
- This is an individual assignment.
- No plagiarism: you are encouraged to do reseach, however, do your own work. Do not copy-and-paste other people's work.

Submission:

- Submit this notebook file and the pdf version - remember to add your name in the filename.
- Deadline: 11:59 pm, 12/8 (Wednesday)

The PIMA Diabetic Data Set

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. It consists of several diagnostic measurements from female patients at least 21 years old of Pima Indian heritage. It also shows the diagnosis on whether the patients have diabetes mellitus disease.

The filename of the dataset is "diabetes.csv" that comes with this assignment.

The dataset contains the following features/columns:

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration at 2 hour in an oral glucose tolerance test (mg/dL)
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-hour serum insulin level (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- DiabetesPedigreeFunction: a function which scores likelihood of diabetes based on family history
- Age: age of patients (years)
- Outcome: class variable 0 or 1 indicating disease (0: non-diabetic, 1: diabetic)

Part 1: Data Preparation and Cleaning (10 points)

Some typical tasks in this part include:

1. Load the dataset in a data frame
2. Examine the dataset attributes: index, columns, range of values etc.
3. Handle missing and invalid data

Note: Mandatory work on handling missing data:

Q1: Are there missing values in the data set?

Q2: You may notice some of the columns have unreasonable zero values (for example, Glucose and BMI). Identify those columns and replace the zeros with the median value of that column.

In [160]:
```
#import pandas

import pandas as pd

#import diabetes.csv as data frame

diabetes = pd.read_csv("C:/Users/lseam/Downloads/Final Data Analysis Assignment

#sum all missing values and print result

print("There are", diabetes.isnull().sum().sum(), "missing values")


#identify columns with unreasonable zero values and replace with median values

diabetes['Glucose'] = diabetes['Glucose'].replace(0,diabetes['Glucose'].median())

diabetes['BloodPressure'] = diabetes['BloodPressure'].replace(0,diabetes['BloodPr

diabetes['SkinThickness'] = diabetes['SkinThickness'].replace(0,diabetes['SkinThi

diabetes['BMI'] = diabetes['BMI'].replace(0,diabetes['BMI'].median())

diabetes['Insulin'] = diabetes['Insulin'].replace(0,diabetes['Insulin'].median())

#print first ten rows of dataframe

print(diabetes.head(10))
```

```
There are 0 missing values
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35     30.5  33.6
1            1       85             66             29     30.5  26.6
2            8      183             64             23     30.5  23.3
3            1       89             66             23     94.0  28.1
4            0      137             40             35    168.0  43.1
5            5      116             74             23     30.5  25.6
6            3       78             50             32     88.0  31.0
7           10      115             72             23     30.5  35.3
8            2      197             70             45    543.0  30.5
9            8      125             96             23     30.5  32.0

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
5                     0.201   30        0
```

6	0.248	26	1
7	0.134	29	0
8	0.158	53	1
9	0.232	54	1

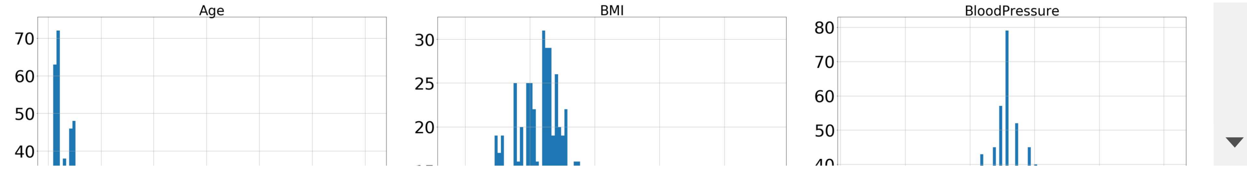
Part 2: Exploratory Data Analysis and Visualization (20 points)

You are expected to perform some basic data analysis and create **at least 4 different charts**. Some examples are:

- Distribution of numeric columns using histogram or bar charts;
- Relationship between column data using scatter plots or pairplot.

Please make comments on the insights from the exploratory analysis.

In [180]:
```
import matplotlib.pyplot as plt

#create histograms for all columns showing distribution, define default graph siz

#Age, Diabetes Pedigree Function, Insulin, Outcome, Pregnancies, and Skin Thickne

#BMI is slightly skewed right and Glucose is slightly skewed left

#Blood pressure is symmetric

diabetes.hist(bins=100, xlabelsize=50, ylabelsize=50, figsize=(50, 40))



#tidy up the layout

plt.tight_layout()


#create several scatter plots: Age vs. Diabetes Pedigree Function, Glucose vs. Di
#and BMI vs. Insulin
#There seems to be weak correlation between variables in all the scatter plots



diabetes.plot.scatter(x = 'Age', y = 'DiabetesPedigreeFunction', c = 'red')

diabetes.plot.scatter(x = 'Glucose', y = 'DiabetesPedigreeFunction', c = 'green')

diabetes.plot.scatter(x = 'Insulin', y = 'DiabetesPedigreeFunction', c = 'blue')

diabetes.plot.scatter(x = 'BMI', y = 'Insulin', c = 'black')


plt.show()
```



Part 3: In-Depth Analysis (25 points)

In this section, you will come up with **at least three interesting questions** about the dataset and write codes to answer the questions. For example, you may analyze how individual feature (column data) impacts the outcome of the diagnosis.

Some example questions:

1. Do older women have higher chances of getting diabetes? You may need to create a bar chart with women in different age groups and show the percentage and/or total number of diabetic vs. non-diabetic in each group.
2. Based on BMI data, how many of this group of patients are considered underweight, normal, overweight, obese (class I, II, and III)? You need to research to find out the definitions.
3. Does high glucose level mean high risk for diabetes? You may want to analyze the relationship between glucose level and the diagnosis outcome.

In [163]:
```
#import linear regression


#Question 1:  How many of this group of patients are considered underweight, norm

#create variables for four categories

underweight, normal, overweight, obese = 0, 0, 0, 0

for value in diabetes['BMI']:  #iterate through BMI and count how many people fit

    if value < 18.5:

        underweight += 1

    elif value >= 18.5 and value <= 24.9:

        normal += 1

    elif value >= 25.0 and value <= 29.9:

        overweight += 1

    elif value >= 30.0:

        obese +=1

#print results

print("\nThere are",underweight, "underweight people in this dataset")

print("\nThere are", normal, "normal people in this dataset")

print("\nThere are", overweight, "overweight people in this dataset")

print("\nThere are", obese, "obese people in this dataset")


#Question 2.  Does higher blood glucose level mean a higher risk of diabetes?

pearson = diabetes['Glucose'].corr(diabetes['Outcome'])

print("\nThe Pearson correlation coefficient is", pearson, "so there is a moderat


#Question 3.  Does older age make it more likely for a person to have higher insu

pearson2 = diabetes['Age'].corr(diabetes['DiabetesPedigreeFunction'])

print("\nThe Pearson correlation coefficient for Age and Diabetes Pedigree Functi
```

```
There are 4 underweight people in this dataset

There are 102 normal people in this dataset

There are 179 overweight people in this dataset

There are 483 obese people in this dataset

The Pearson correlation coefficient is 0.4927824039150267 so there is a moder
ate positive correlation between glucose level and diabetes outcome

The Pearson correlation coefficient for Age and Diabetes Pedigree Function is
0.033561312434805514 so there is no linear relationship between the two
```

Part 4: Regression Model and Outcome Prediction (45 points)

In this section, you will build a diabetic outcome prediction model based on **Logistic Regression**.

Step 1: Train-Test Split

Split the dataset into training set and testing set using proper Sklearn package. Use a test size of 30% and your own arbitrary number for random_state.

Since we want to classify whether a patient is diabetic or not, the output(target) will be the "Outcome" column. The rest of the columns will be the input (features). So you will drop the Outcome column from the dataset and make it X and pick the Outcome column and make it y.

In [175]:
```
#import sklearn modules


from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression

#create our x and y values

y = diabetes['Outcome']

x = diabetes.drop(['Outcome'], axis=1)


#split our dataset

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_s

#print first five rows of our split dataset

print("x_ train is \n", x_train.head(), "\n \n", "x_test is \n", x_test.head(),"\
```

```
x_ train is
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
580            0      151             90             46     30.5  42.1
418            1       83             68             23     30.5  18.2
764            2      122             70             27     30.5  36.8
363            4      146             78             23     30.5  38.5
757            0      123             72             23     30.5  36.3

     DiabetesPedigreeFunction  Age
580                     0.371   21
418                     0.624   27
764                     0.340   27
363                     0.520   67
757                     0.258   52

 x_test is
      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
661            1      199             76             43     30.5  42.9
122            2      107             74             30    100.0  33.6
113            4       76             62             23     30.5  34.0
14             5      166             72             19    175.0  25.8
529            0      111             65             23     30.5  24.6

     DiabetesPedigreeFunction  Age
661                     1.394   22
122                     0.404   23
113                     0.391   25
```

```
14                       0.587   51
529                      0.660   31

 y_train is
 580    1
418    0
764    0
363    1
757    1
Name: Outcome, dtype: int64

 y_test is
 661    1
122    0
113    0
14     1
529    0
Name: Outcome, dtype: int64
```

Step 2: Train and Fit Your Model

Use Sklearn package on the training data and obtain the prediction model.

In [177]:

```
#define our logistic regression model

model = LogisticRegression()



#fit our model

model.fit(x_train, y_train)

#create our prediction model

predictions = (model.predict(x_test))
```

```
C:\Users\lseam\AppData\Roaming\Python\Python36\site-packages\sklearn\linear_mod
el\_logistic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Step 3: Prediction and Model Evaluation

3.1. Use Sklearn package to predict the outcomes on the test data.

In [178]:

```
#print predictions

print(predictions)
```

```
[1 0 0 1 0 0 1 1 1 0 1 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0
 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1
 1 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 1 0 0 0 1 0
 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0
 0 0 0 1 0 0 1 0 1 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1
 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
 1 0 0 1 0 0 0 0 0]
```

3.2. Use Sklearn package to create the Confusion Matrix.

What are the values of TP (True Positive), TN (True Negative), FP (False Positive) and FN (False Negative)?

In [167]:
```
#import confusion matrix and classification report modules

from sklearn.metrics import confusion_matrix

from sklearn.metrics import classification_report

#create and print our confusion matrix

confusionMatrix = confusion_matrix(y_test, predictions, labels=[1,0])

print(confusionMatrix)

#print the individual values

print("TP is", confusionMatrix[0,0])

print("TN is", confusionMatrix[1,1])

print("FP is", confusionMatrix[1,0])

print("FN is", confusionMatrix[0,1])
```

```
[[ 38  36]
 [ 18 139]]
TP is 38
TN is 139
FP is 18
FN is 36
```

3.3. Use Sklearn package to create the Classification Report.

What are the values of Accuracy, Precision, Recall, and Sensitivity? (You may want to refer back to the lecture note for the definition)

In [169]:
```
#calculate and display classification report

#create and print our classification report

report = classification_report(y_test, predictions)

print(report)

#calculate and print the accuracy, precision, sensitivity/recall

print("The accuracy is", ((confusionMatrix[0,0])+(confusionMatrix[1,1]))/len(pred

print('The precision is', (confusionMatrix[0,0])/(confusionMatrix[0,0]+confusionM

print('The sensitivity/recall is', (confusionMatrix[0,0])/(confusionMatrix[0,0]+c
```

```
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       157
           1       0.68      0.51      0.58        74

    accuracy                           0.77       231
   macro avg       0.74      0.70      0.71       231
weighted avg       0.76      0.77      0.76       231

The accuracy is 0.7662337662337663
The precision is 0.6785714285714286
The sensitivity/recall is 0.5135135135135135
```

Step 4: Interpreting the Prediction Model

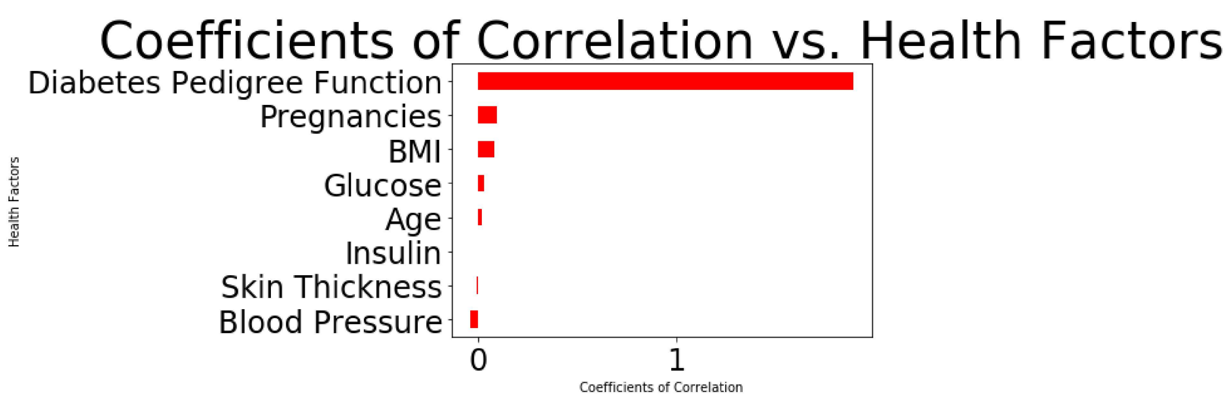
To get a better sense of the logistic regression model, we will retrieve the regression coefficients and visualize which features have greater impact on the prediction outcome.

4.1. Follow the instruction:

1. Use "coeff = list(lr*model.coef*[0])" to obtain the regression coefficients, where lr_model is the name of your model.
2. Make a horizontal bar chart with the feature lables (i.e., Glucose, BMI, etc.) in the y-axis and the coefficients in the x-axis. Order the coefficents in descending order.

In [159]:

```
#creating index for our series

label =  ['Pregnancies','Glucose','Blood Pressure','Skin Thickness','Insulin','BM

#obtain our regression coefficients

coeff = list(model.coef_[0])

#create pandas series

coeff_series = pd.Series(coeff, index=label)

#sort in descending order

coeff_desc = coeff_series.sort_values()

#create and show horizontal bar graph

ax = coeff_desc.plot.barh(x='Coefficient of Correlation', y='Health Factor', colo

ax.set_xlabel('Coefficients of Correlation')

ax.set_ylabel('Health Factors')

ax.set_title('Coefficients of Correlation vs. Health Factors')

plt.show()
```



4.2. Based on the observation on the coefficients and the chart, please answer the following questions:

Q1: What are the top three factors that have significant influence on the prediction outcome?

Q2: Do those three factors also have high correlation coefficients with the outcome?

In [156]:
```
#Q1

#print the last three items in series ie., top three factors

print("The top three factors are: \n", coeff_desc.tail(3))

#Q2

#print the last item in the series, the only one with coefficient >> 0.5

print("Only", coeff_desc.index[len(coeff_desc)-1], "with a value of", coeff_desc.
```

```
The top three factors are:
 BMI                           0.080877
Pregnancies                   0.095594
Diabetes Pedigree Function    1.888639
dtype: float64
Only  Diabetes Pedigree Function with a value of 1.8886386282388927 has a high
correlation coefficient
```

Wonderful, you are done! It has been a fun semester. Happy Holiday!