Healthcare PGP

DESCRIPTION

Problem Statement

NIDDK (National Institute of Diabetes and Digestive and Kidney Diseases) research creates knowledge about and treatments for the most chronic, costly, and consequential diseases. The dataset used in this project is originally from NIDDK. The objective is to predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Build a model to accurately predict whether the patients in the dataset have diabetes or not.

Dataset Description

The datasets consists of several medical predictor variables and one target variable (Outcome). Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and more.

Variables	Description
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration in an oral glucose tolerance test
BloodPressure	Diastolic blood pressure (mm Hg)
SkinThickness	Triceps skinfold thickness (mm)
Insulin	Two hour serum insulin
BMI	Body Mass Index
DiabetesPedigreeFunction	Diabetes pedigree function
Age	Age in years
Outcome	Class variable (either 0 or 1). 268 of 768 values are 1, and the others are 0

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
import os
os.chdir(r'D:\Study\PGP in Data Science\Project_2\Project 2\Healthcare - Diabetes')
```

In [3]:
```
patient_data = pd.read_csv('health_care_diabetes.csv')
```

In [4]:
```
patient_data.head()
```

Out[4]:

	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

In [5]:
```
patient_data.describe()
```

Out[5]:

	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	768.00000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.34895
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.47695
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.00000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.00000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.00000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.00000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.00000

Project Task: Week 1

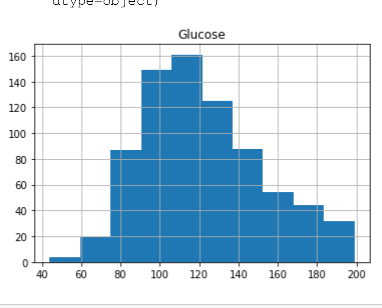
Data Exploration:

1. Perform descriptive analysis. Understand the variables and their corresponding values. On the columns below, a value of zero does not make sense and thus indicates missing value:
   - Glucose
   - BloodPressure
   - SkinThickness
   - Insulin
   - BMI
2. Visually explore these variables using histograms. Treat the missing values accordingly.
3. There are integer and float data type variables in this dataset. Create a count (frequency) plot describing the data types and the count of variables.

In [6]:
```
glucose_notnull_data = patient_data[patient_data['Glucose']!=0]
```

In [7]:
```
glucose_notnull_data.hist(column='Glucose')
```

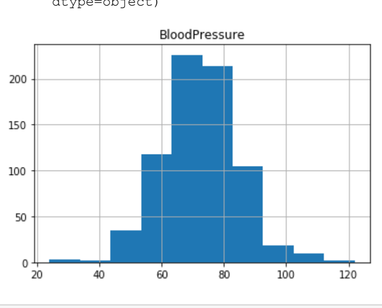
Out[7]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000154CFE378D0>]],
      dtype=object)
```



In [8]:
```
bloodpressure_notnull_data = patient_data[patient_data['BloodPressure']!=0]
```

In [9]:
```
bloodpressure_notnull_data.hist(column='BloodPressure')
```

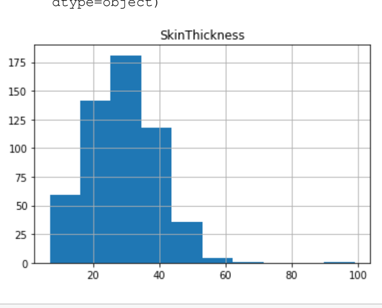
Out[9]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000154D01A28D0>]],
      dtype=object)
```



In [10]:
```
skinthickness_notnull_data = patient_data[patient_data['SkinThickness']!=0]
```

In [11]:
```
skinthickness_notnull_data.hist(column='SkinThickness')
```

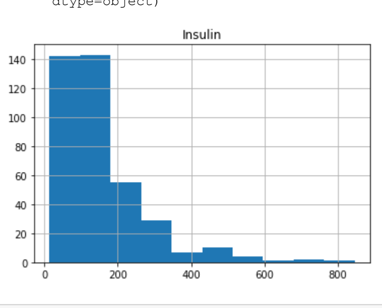
Out[11]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000154D0243FD0>]],
      dtype=object)
```



In [12]:
```
insulin_notnull_data = patient_data[patient_data['Insulin']!=0]
```

In [13]:
```
insulin_notnull_data.hist(column='Insulin')
```

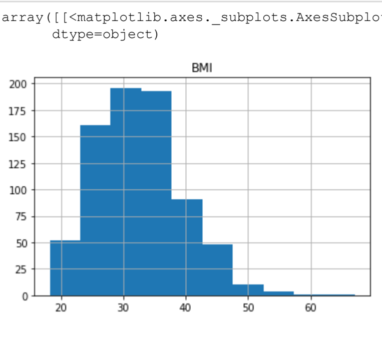
Out[13]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000154D02BC240>]],
      dtype=object)
```



In [14]:
```
BMI_notnull_data = patient_data[patient_data['BMI']!=0]
```

In [15]:
```
BMI_notnull_data.hist(column='BMI')
```

Out[15]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000154D033D080>]],
      dtype=object)
```



INFERENCE:

1. Data for Glucose seems to be equally distributed. So taking mean for replacing 0 values should be the best approach
2. Data for BloodPressure seems to be equally distributed. So taking mean for replacing 0 values should be the best approach
3. Data for SkinThickness seems to be left skewed. So taking median for replacing 0 values should be the best approach
4. Data for Insulin seems to be highly left skewed. So taking mode for replacing 0 values should be the best approach
5. Data for BMI seems to be left skewed. So taking median for replacing 0 values should be the best approach

In [16]:
```
patient_data.loc[patient_data['Glucose'] == 0,'Glucose'] = int(glucose_notnull_data['Glucose'].mean())
patient_data.loc[patient_data['BloodPressure'] == 0,'BloodPressure'] = int(bloodpressure_notnull_data[
'BloodPressure'].mean())
patient_data.loc[patient_data['SkinThickness'] == 0,'SkinThickness'] = int(skinthickness_notnull_data[
'SkinThickness'].median())
patient_data.loc[patient_data['Insulin'] == 0,'Insulin'] = int(insulin_notnull_data['Insulin'].mode())
patient_data.loc[patient_data['BMI'] == 0,'BMI'] = BMI_notnull_data['BMI'].median()
```

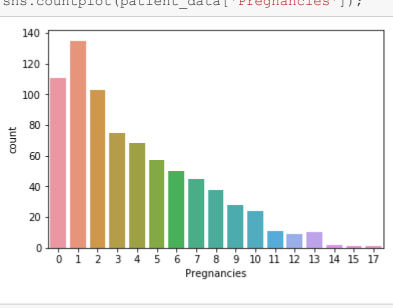
In [17]:
```
patient_data.head()
```

Out[17]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	105	33.6	0.627	50	1
1	1	85	66	29	105	26.6	0.351	31	0
2	8	183	64	29	105	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

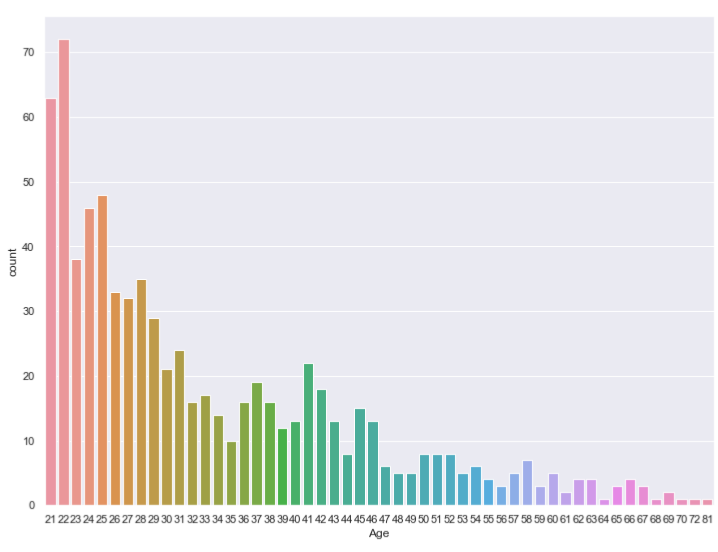
In [18]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [19]:
```
sns.countplot(patient_data['Pregnancies']);
```



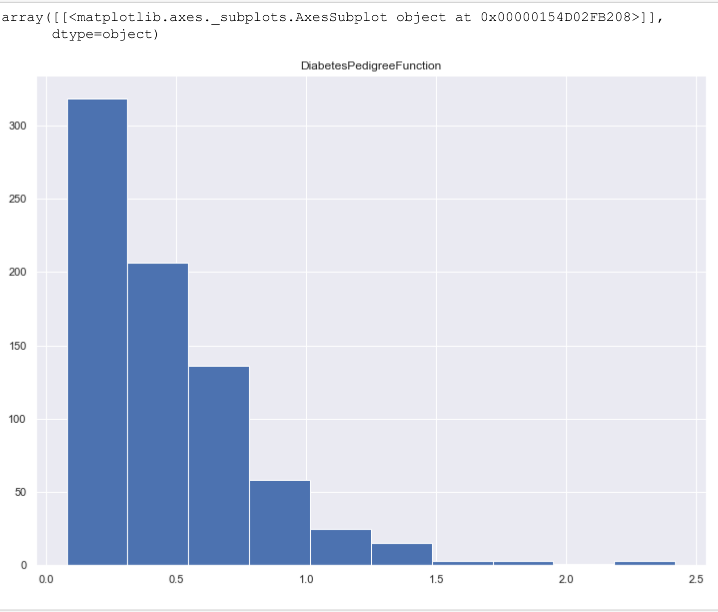
In [20]:
```
sns.set(rc={'figure.figsize':(12,9)})
sns.countplot(patient_data['Age'])
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x154d224f048>
```



In [21]:
```
patient_data.hist(column='DiabetesPedigreeFunction')
```

Out[21]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000154D02FB208>]],
      dtype=object)
```



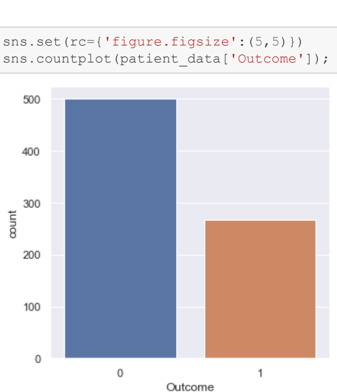
In [22]:
```
patient_data.to_csv('healthcare_data_preprocessing_done.csv',index=False)
```

Project Task: Week 2

Data Exploration:

1. Check the balance of the data by plotting the count of outcomes by their value. Describe your findings and plan future course of action.
2. Create scatter charts between the pair of variables to understand the relationships. Describe your findings.
3. Perform correlation analysis. Visually explore it using a heat map.

In [23]:
```
sns.set(rc={'figure.figsize':(5,5)})
sns.countplot(patient_data['Outcome']);
```



In [24]:
```
patient_data['Outcome'].value_counts()
```

Out[24]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

INFERENCE:-

> If our model categorizes all outcomes as 0, then it will have an accuracy of 500/768 = 78%. Hence, our model should aim at better classification than 78%

Scatter Plot of Pregnancies vs Other Columns

In [25]:
```
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(241)
plt.scatter(patient_data['Pregnancies'], patient_data['Glucose'])
ax.set_title('Glucose')
ax = fig.add_subplot(242)
plt.scatter(patient_data['Pregnancies'], patient_data['BloodPressure'])
ax.set_title('BloodPressure')
ax = fig.add_subplot(243)
plt.scatter(patient_data['Pregnancies'], patient_data['SkinThickness'])
ax.set_title('SkinThickness')
ax = fig.add_subplot(244)
plt.scatter(patient_data['Pregnancies'], patient_data['Insulin'])
ax.set_title('Insulin')
ax = fig.add_subplot(245)
plt.scatter(patient_data['Pregnancies'], patient_data['BMI'])
ax.set_title('BMI')
ax = fig.add_subplot(246)
plt.scatter(patient_data['Pregnancies'], patient_data['DiabetesPedigreeFunction'])
ax.set_title('DiabetesPedigreeFunction')
ax = fig.add_subplot(247)
plt.scatter(patient_data['Pregnancies'], patient_data['Age'])
ax.set_title('Age')
ax = fig.add_subplot(248)
plt.scatter(patient_data['Pregnancies'], patient_data['Outcome'])
ax.set_title('Outcome')
fig.text(0.06, 0.5, 'Pregnancies', ha='center', va='center', rotation='vertical')
```

Out[25]:
```
Text(0.06, 0.5, 'Pregnancies')
```

Scatter Plot of Glucose vs Other Columns

In [26]:
```
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(241)
plt.scatter(patient_data['Glucose'], patient_data['Pregnancies'])
ax.set_title('Pregnancies')
ax = fig.add_subplot(242)
plt.scatter(patient_data['Glucose'], patient_data['BloodPressure'])
ax.set_title('BloodPressure')
ax = fig.add_subplot(243)
plt.scatter(patient_data['Glucose'], patient_data['SkinThickness'])
ax.set_title('SkinThickness')
ax = fig.add_subplot(244)
plt.scatter(patient_data['Glucose'], patient_data['Insulin'])
ax.set_title('Insulin')
ax = fig.add_subplot(245)
plt.scatter(patient_data['Glucose'], patient_data['BMI'])
ax.set_title('BMI')
ax = fig.add_subplot(246)
plt.scatter(patient_data['Glucose'], patient_data['DiabetesPedigreeFunction'])
ax.set_title('DiabetesPedigreeFunction')
ax = fig.add_subplot(247)
plt.scatter(patient_data['Glucose'], patient_data['Age'])
ax.set_title('Age')
ax = fig.add_subplot(248)
plt.scatter(patient_data['Glucose'], patient_data['Outcome'])
ax.set_title('Outcome')
fig.text(0.06, 0.5, 'Glucose', ha='center', va='center', rotation='vertical')
```

Out[26]:
```
Text(0.06, 0.5, 'Glucose')
```

Scatter Plot of BloodPressure vs Other Columns

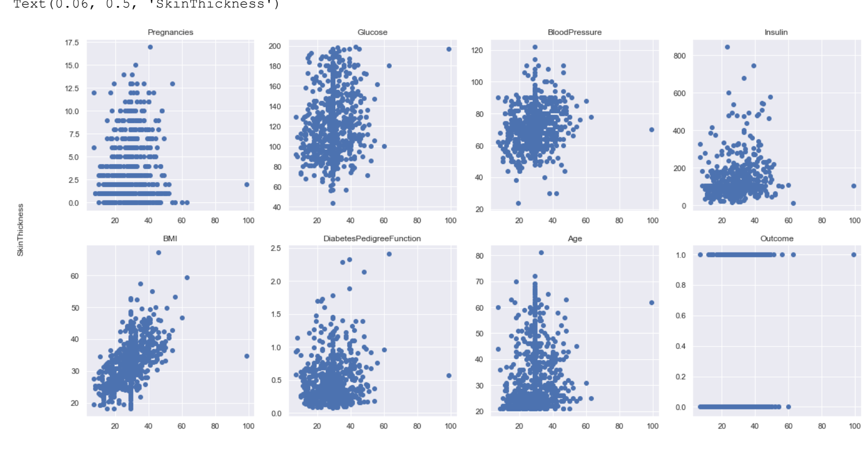
In [27]:
```
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(241)
plt.scatter(patient_data['BloodPressure'], patient_data['Pregnancies'])
ax.set_title('Pregnancies')
ax = fig.add_subplot(242)
plt.scatter(patient_data['BloodPressure'], patient_data['Glucose'])
ax.set_title('Glucose')
ax = fig.add_subplot(243)
plt.scatter(patient_data['BloodPressure'], patient_data['SkinThickness'])
ax.set_title('SkinThickness')
ax = fig.add_subplot(244)
plt.scatter(patient_data['BloodPressure'], patient_data['Insulin'])
ax.set_title('Insulin')
ax = fig.add_subplot(245)
plt.scatter(patient_data['BloodPressure'], patient_data['BMI'])
ax.set_title('BMI')
ax = fig.add_subplot(246)
plt.scatter(patient_data['BloodPressure'], patient_data['DiabetesPedigreeFunction'])
ax.set_title('DiabetesPedigreeFunction')
ax = fig.add_subplot(247)
plt.scatter(patient_data['BloodPressure'], patient_data['Age'])
ax.set_title('Age')
ax = fig.add_subplot(248)
plt.scatter(patient_data['BloodPressure'], patient_data['Outcome'])
ax.set_title('Outcome')
fig.text(0.06, 0.5, 'BloodPressure', ha='center', va='center', rotation='vertical')
```

Out[27]:
```
Text(0.06, 0.5, 'BloodPressure')
```

Scatter Plot of SkinThickness vs Other Columns

In [28]:
```
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(241)
plt.scatter(patient_data['SkinThickness'], patient_data['Pregnancies'])
ax.set_title('Pregnancies')
ax = fig.add_subplot(242)
plt.scatter(patient_data['SkinThickness'], patient_data['Glucose'])
ax.set_title('Glucose')
ax = fig.add_subplot(243)
plt.scatter(patient_data['SkinThickness'], patient_data['BloodPressure'])
ax.set_title('BloodPressure')
ax = fig.add_subplot(244)
plt.scatter(patient_data['SkinThickness'], patient_data['Insulin'])
ax.set_title('Insulin')
ax = fig.add_subplot(245)
plt.scatter(patient_data['SkinThickness'], patient_data['BMI'])
ax.set_title('BMI')
ax = fig.add_subplot(246)
plt.scatter(patient_data['SkinThickness'], patient_data['DiabetesPedigreeFunction'])
ax.set_title('DiabetesPedigreeFunction')
ax = fig.add_subplot(247)
plt.scatter(patient_data['SkinThickness'], patient_data['Age'])
ax.set_title('Age')
ax = fig.add_subplot(248)
plt.scatter(patient_data['SkinThickness'], patient_data['Outcome'])
ax.set_title('Outcome')
fig.text(0.06, 0.5, 'SkinThickness', ha='center', va='center', rotation='vertical')
```

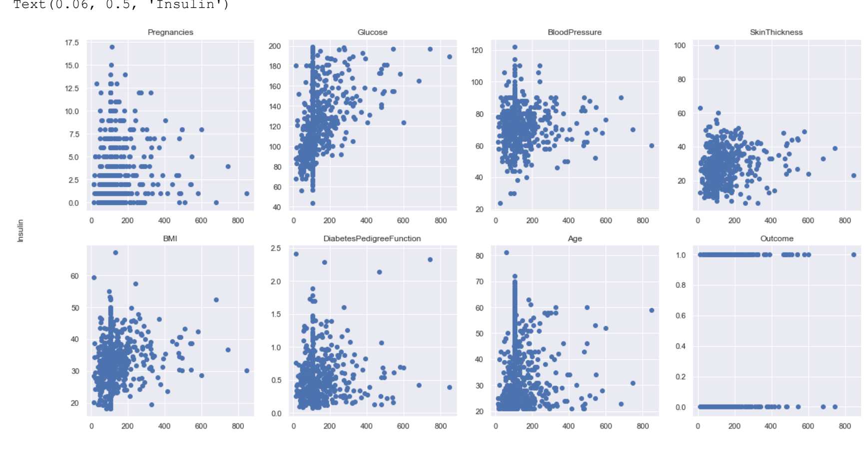
Out[28]:
```
Text(0.06, 0.5, 'SkinThickness')
```



Scatter Plot of Insulin vs Other Columns

In [29]:
```
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(241)
plt.scatter(patient_data['Insulin'], patient_data['Pregnancies'])
ax.set_title('Pregnancies')
ax = fig.add_subplot(242)
plt.scatter(patient_data['Insulin'], patient_data['Glucose'])
ax.set_title('Glucose')
ax = fig.add_subplot(243)
plt.scatter(patient_data['Insulin'], patient_data['BloodPressure'])
ax.set_title('BloodPressure')
ax = fig.add_subplot(244)
plt.scatter(patient_data['Insulin'], patient_data['SkinThickness'])
ax.set_title('SkinThickness')
ax = fig.add_subplot(245)
plt.scatter(patient_data['Insulin'], patient_data['BMI'])
ax.set_title('BMI')
ax = fig.add_subplot(246)
plt.scatter(patient_data['Insulin'], patient_data['DiabetesPedigreeFunction'])
ax.set_title('DiabetesPedigreeFunction')
ax = fig.add_subplot(247)
plt.scatter(patient_data['Insulin'], patient_data['Age'])
ax.set_title('Age')
ax = fig.add_subplot(248)
plt.scatter(patient_data['Insulin'], patient_data['Outcome'])
ax.set_title('Outcome')
fig.text(0.06, 0.5, 'Insulin', ha='center', va='center', rotation='vertical')
```

Out[29]:
```
Text(0.06, 0.5, 'Insulin')
```



Scatter Plot of BMI vs Other Columns

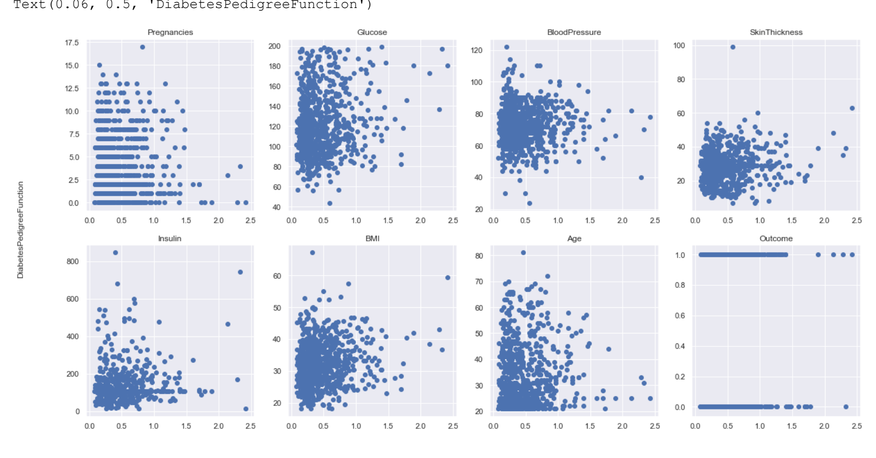
In [30]:
```
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(241)
plt.scatter(patient_data['BMI'], patient_data['Pregnancies'])
ax.set_title('Pregnancies')
ax = fig.add_subplot(242)
plt.scatter(patient_data['BMI'], patient_data['Glucose'])
ax.set_title('Glucose')
ax = fig.add_subplot(243)
plt.scatter(patient_data['BMI'], patient_data['BloodPressure'])
ax.set_title('BloodPressure')
ax = fig.add_subplot(244)
plt.scatter(patient_data['BMI'], patient_data['SkinThickness'])
ax.set_title('SkinThickness')
ax = fig.add_subplot(245)
plt.scatter(patient_data['BMI'], patient_data['Insulin'])
ax.set_title('Insulin')
ax = fig.add_subplot(246)
plt.scatter(patient_data['BMI'], patient_data['DiabetesPedigreeFunction'])
ax.set_title('DiabetesPedigreeFunction')
ax = fig.add_subplot(247)
plt.scatter(patient_data['BMI'], patient_data['Age'])
ax.set_title('Age')
ax = fig.add_subplot(248)
plt.scatter(patient_data['BMI'], patient_data['Outcome'])
ax.set_title('Outcome')
fig.text(0.06, 0.5, 'BMI', ha='center', va='center', rotation='vertical')
```

Out[30]:
```
Text(0.06, 0.5, 'BMI')
```

Scatter Plot of Diabetes Pedigree Function vs Other Columns

In [31]:
```
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(241)
plt.scatter(patient_data['DiabetesPedigreeFunction'], patient_data['Pregnancies'])
ax.set_title('Pregnancies')
ax = fig.add_subplot(242)
plt.scatter(patient_data['DiabetesPedigreeFunction'], patient_data['Glucose'])
ax.set_title('Glucose')
ax = fig.add_subplot(243)
plt.scatter(patient_data['DiabetesPedigreeFunction'], patient_data['BloodPressure'])
ax.set_title('BloodPressure')
ax = fig.add_subplot(244)
plt.scatter(patient_data['DiabetesPedigreeFunction'], patient_data['SkinThickness'])
ax.set_title('SkinThickness')
ax = fig.add_subplot(245)
plt.scatter(patient_data['DiabetesPedigreeFunction'], patient_data['Insulin'])
ax.set_title('Insulin')
ax = fig.add_subplot(246)
plt.scatter(patient_data['DiabetesPedigreeFunction'], patient_data['BMI'])
ax.set_title('BMI')
ax = fig.add_subplot(247)
plt.scatter(patient_data['DiabetesPedigreeFunction'], patient_data['Age'])
ax.set_title('Age')
ax = fig.add_subplot(248)
plt.scatter(patient_data['DiabetesPedigreeFunction'], patient_data['Outcome'])
ax.set_title('Outcome')
fig.text(0.06, 0.5, 'DiabetesPedigreeFunction', ha='center', va='center', rotation='vertical')
```

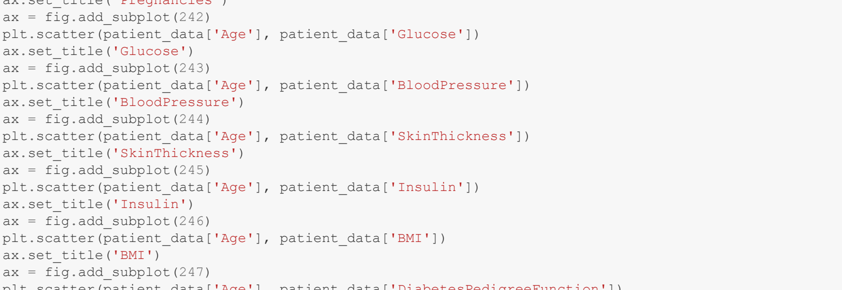
Out[31]:
```
Text(0.06, 0.5, 'DiabetesPedigreeFunction')
```



Scatter Plot of Age vs Other Columns

In [32]:
```python
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(241)
plt.scatter(patient_data['Age'], patient_data['Pregnancies'])
ax.set_title('Pregnancies')
ax = fig.add_subplot(242)
plt.scatter(patient_data['Age'], patient_data['Glucose'])
ax.set_title('Glucose')
ax = fig.add_subplot(243)
plt.scatter(patient_data['Age'], patient_data['BloodPressure'])
ax.set_title('BloodPressure')
ax = fig.add_subplot(244)
plt.scatter(patient_data['Age'], patient_data['SkinThickness'])
ax.set_title('SkinThickness')
ax = fig.add_subplot(245)
plt.scatter(patient_data['Age'], patient_data['Insulin'])
ax.set_title('Insulin')
ax = fig.add_subplot(246)
plt.scatter(patient_data['Age'], patient_data['BMI'])
ax.set_title('BMI')
ax = fig.add_subplot(247)
plt.scatter(patient_data['Age'], patient_data['DiabetesPedigreeFunction'])
ax.set_title('DiabetesPedigreeFunction')
ax = fig.add_subplot(248)
plt.scatter(patient_data['Age'], patient_data['Outcome'])
ax.set_title('Outcome')
fig.text(0.06, 0.5, 'Age', ha='center', va='center', rotation='vertical')
```

Out[32]:
```
Text(0.06, 0.5, 'Age')
```



INFERENCE:

1. Pregancies seems to be directly related to Age
2. Insulin seems to be directly related to Glucose
3. BMI seems to be directly related to SkinThickness
4. There doesn't seem to be much correlation between other columns

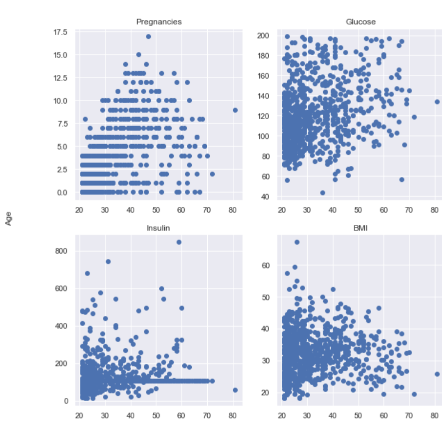
CORRELATION MATRIX

In [33]:
```python
corrMatrix = patient_data.corr()
corrMatrix
```

Out[33]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
Pregnancies	1.000000	0.127957	0.208615	0.081770	0.005204	0.021559	-0.033523	0.544341
Glucose	0.127957	1.000000	0.218615	0.192677	0.411711	0.231119	0.137100	0.266591
BloodPressure	0.208615	0.218615	1.000000	0.191892	0.027595	0.281257	-0.002378	0.324915
SkinThickness	0.081770	0.192677	0.191892	1.000000	0.151525	0.543205	0.102188	0.126107
Insulin	0.005204	0.411711	0.027595	0.151525	1.000000	0.185899	0.141959	0.070669
BMI	0.021559	0.231119	0.281257	0.543205	0.185899	1.000000	0.153438	0.025597
DiabetesPedigreeFunction	-0.033523	0.137100	-0.002378	0.102188	0.141959	0.153438	1.000000	0.033561
Age	0.544341	0.266591	0.324915	0.126107	0.070669	0.025597	0.033561	1.000000
Outcome	0.221898	0.492911	0.165723	0.214873	0.193850	0.312038	0.173844	0.238356

In [34]:
```python
sns.heatmap(corrMatrix, annot=True)
plt.show()
```



INFERENCE:

> The Heatmap and correlation matrix confirms our suspicions with the scatter plots regarding the correlation between the columns. One additional observation is that the outcome also seems to be directly related to the Glucose levels

Project Task: Week 3

Data Modeling:

1. Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.
2. Apply an appropriate classification algorithm to build a model. Compare various models with the results from KNN algorithm.
3. Create a classification report by analyzing sensitivity, specificity, AUC (ROC curve), etc. Please be descriptive to explain what values of these parameter you have used.

In [35]:
```python
#DIVIDING DATA INTO TRAIN AND TEST SETS
from sklearn.model_selection import train_test_split as split
train, test = split(patient_data, train_size = 0.8, random_state = 123)
```

In [36]:
```python
#EXTRACTING X AND Y FROM DATA
x_train = train.drop('Outcome', axis = 1)
y_train = train.Outcome
x_test = test.drop('Outcome', axis = 1)
y_test = test.Outcome
```

In [37]:
```python
feature_names = patient_data.drop('Outcome',axis=1)
feature_names.head()
```

Out[37]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	105	33.6	0.627	50
1	1	85	66	29	105	26.6	0.351	31
2	8	183	64	29	105	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

Strategy Used in selecting Classifier Models for our Prediction

Classification is a subcategory of supervised learning where the goal is to predict the categorical class labels (discrete, unoredered values, group membership) of new instances based on past observations. Our aim is to create separate models and compare them on the basis of their accuracy and F1-score as well as other parameters such as sensitivity and specificity.

For our Study, we'll be using the following 3 models:-

1. Random Forest Classifier
2. Support Vector Machine Classifier
3. K - Nearest Neighbours Classifier

Model 1: Random Forest Classifier

Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.
Reasons for using Random Forest Classifier:-

1. Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.
2. It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.

In [38]:
```python
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
rfclf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
rfclf.fit(x_train,y_train)
```

Out[38]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [39]:
```python
y_pred_rcf=rfclf.predict(x_test)
```

In [40]:
```python
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred_rcf))
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred_rcf))
# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred_rcf))
```

```
Accuracy: 0.7987012987012987
Precision: 0.7547169811320755
Recall: 0.6896551724137931
```

Model 2: Support Vector Machine Classifier

SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes.
Reasons for using SVM classifier:-

1. SVM Classifiers offer good accuracy and perform faster prediction compared to Naïve Bayes algorithm.
2. They also use less memory because they use a subset of training points in the decision phase.

In [41]:
```python
#Import svm model
from sklearn import svm

#Create a svm Classifier
svmclf = svm.SVC(kernel='linear') #Linear Kernel

#Train the model using the training sets
svmclf.fit(x_train, y_train)

#Predict the response for test dataset
y_pred_svm = svmclf.predict(x_test)
```

In [42]:
```python
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred_svm))
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred_svm))
# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred_svm))
```

```
Accuracy: 0.7922077922077922
Precision: 0.7954545454545454
Recall: 0.603448275862069
```

Model 3: K-Nearest Neighbours Classifier

K Nearest Neighbor(KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms.
Reasons for using KNN Classifier:-

1. The training phase of K-nearest neighbor classification is much faster compared to other classification algorithms.
2. There is no need to train a model for generalization, That is why KNN is known as the simple and instance-based learning algorithm.

In [43]:
```python
#Import knearest neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier

#Create KNN Classifier
knn = KNeighborsClassifier(n_neighbors=2)

#Train the model using the training sets
knn.fit(x_train, y_train)

#Predict the response for test dataset
y_pred_knn = knn.predict(x_test)
```

In [44]:
```python
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred_knn))
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred_knn))
# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred_knn))
```

```
Accuracy: 0.7402597402597403
Precision: 0.8461538461538461
Recall: 0.3793103448275862
```

In [45]:
```python
from sklearn.metrics import classification_report
print("RCF CLASSIFICATION REPORT : ")
print(classification_report(y_test, y_pred_rcf))
print("SVM CLASSIFICATION REPORT : ")
print(classification_report(y_test, y_pred_svm))
print("KNN CLASSIFICATION REPORT : ")
print(classification_report(y_test, y_pred_knn))
```

```
RCF CLASSIFICATION REPORT :
              precision    recall  f1-score   support

           0       0.82      0.86      0.84        96
           1       0.75      0.69      0.72        58

    accuracy                           0.80       154
   macro avg       0.79      0.78      0.78       154
weighted avg       0.80      0.80      0.80       154

SVM CLASSIFICATION REPORT :
              precision    recall  f1-score   support

           0       0.79      0.91      0.84        96
           1       0.80      0.60      0.69        58

    accuracy                           0.79       154
   macro avg       0.79      0.75      0.77       154
weighted avg       0.79      0.79      0.79       154

KNN CLASSIFICATION REPORT :
              precision    recall  f1-score   support

           0       0.72      0.96      0.82        96
           1       0.85      0.38      0.52        58

    accuracy                           0.74       154
   macro avg       0.78      0.67      0.67       154
weighted avg       0.77      0.74      0.71       154
```

INFERENCE:

> Random Forest and Support Vector Machine Classifiers perform better than the KNN Classifier based on the Accuracy and F1-score

image

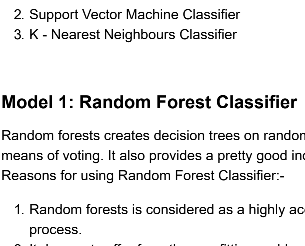
In [46]:
```python
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

In [48]:
```python
rcf_auc = roc_auc_score(y_test, y_pred_rcf)
svm_auc = roc_auc_score(y_test, y_pred_svm)
knn_auc = roc_auc_score(y_test, y_pred_knn)
# summarize scores
print('RCF: ROC AUC=%.3f' % (rcf_auc))
print('SVM: ROC AUC=%.3f' % (svm_auc))
print('KNN: ROC AUC=%.3f' % (knn_auc))
```

```
RCF: ROC AUC=0.777
SVM: ROC AUC=0.755
KNN: ROC AUC=0.669
```

In [49]:
```python
# calculate roc curves
rcf_fpr, rcf_tpr, _ = roc_curve(y_test, y_pred_rcf)
svm_fpr, svm_tpr, _ = roc_curve(y_test, y_pred_svm)
knn_fpr, knn_tpr, _ = roc_curve(y_test, y_pred_knn)
# plot the roc curve for the model
plt.plot(rcf_fpr, rcf_tpr, marker='.', label='Random Forest Classifier', color='red')
plt.plot(svm_fpr, svm_tpr, marker='.', label='Support Vector Machine', color='blue')
plt.plot(knn_fpr, knn_tpr, marker='.', label='K-Nearest Neighbour', color='green')
# axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

Out[49]:
```
Text(0, 0.5, 'True Positive Rate')
```



In [50]:
```python
from sklearn.metrics import confusion_matrix
tn_rcf, fp_rcf, fn_rcf, tp_rcf = confusion_matrix(y_test, y_pred_rcf).ravel()
tn_svm, fp_svm, fn_svm, tp_svm = confusion_matrix(y_test, y_pred_svm).ravel()
tn_knn, fp_knn, fn_knn, tp_knn = confusion_matrix(y_test, y_pred_knn).ravel()
print ('Confusion matrix for RCF : ')
print (confusion_matrix(y_test, y_pred_rcf))
print ('Confusion matrix for SVM : ')
print (confusion_matrix(y_test, y_pred_svm))
print ('Confusion matrix for KNN : ')
print (confusion_matrix(y_test, y_pred_knn))
```

```
Confusion matrix for RCF :
[[83 13]
 [18 40]]
Confusion matrix for SVM :
[[87  9]
 [23 35]]
Confusion matrix for KNN :
[[92  4]
 [36 22]]
```

In [51]:
```python
specificity_rcf = tn_rcf / (tn_rcf+fp_rcf)
specificity_svm = tn_svm / (tn_svm+fp_svm)
specificity_knn = tn_knn / (tn_knn+fp_knn)
sensitivity_rcf = tp_rcf / (tp_rcf+fn_rcf)
sensitivity_svm = tp_svm / (tp_svm+fn_svm)
sensitivity_knn = tp_knn / (tp_knn+fn_knn)

# Sensitivity scores
print('RCF: Sensitivity=%.2f' % (sensitivity_rcf))
print('SVM: Sensitivity=%.2f' % (sensitivity_svm))
print('KNN: Sensitivity=%.2f' % (sensitivity_knn))


# Specificity scores
print('RCF: Specificity=%.2f' % (specificity_rcf))
print('SVM: Specificity=%.2f' % (specificity_svm))
print('KNN: Specificity=%.2f' % (specificity_knn))
```

```
RCF: Sensitivity=0.69
SVM: Sensitivity=0.60
KNN: Sensitivity=0.38
RCF: Specificity=0.86
SVM: Specificity=0.91
KNN: Specificity=0.96
```

Project Task: Week 4

Data Reporting:

1. Create a dashboard in tableau by choosing appropriate chart types and metrics useful for the business. The dashboard must entail the following:

   a. Pie chart to describe the diabetic or non-diabetic population

   b. Scatter charts between relevant variables to analyze the relationships

   c. Histogram or frequency charts to analyze the distribution of the data

   d. Create bins of these age values: 20-25, 25-30, 30-35, etc. Analyze different variables for these age brackets using a bubble chart.

Pie Chart

Scatter Charts

Histogram of Various Charts

Bubble Chart

Dashboard

The Tableau Project can be found here

In [ ]: