1. Import libraries

import all the modules, functions and objects

In [9]:
```
# libraries
import pandas
import pylab
import numpy as np
import seaborn as sns
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
```

2. Load Dataset

We are goint to use the diabetes dataset from the National Institute of Diabetes and Digestive and Kidney Diseases. We are using pandas to load the data. We will also use pandas next to explore the data both with descriptive statistics and data visualization.

In [10]:
```
dataset = pandas.read_csv('diabetes.csv')
```

In [11]:
```
dataset.head()
```

Out[11]:

	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

3. Summarize the Dataset

3.1 Dimensions of Dataset

In [12]:
```
# shape
print(dataset.shape)
```

```
(768, 9)
```

3.2 Peek at Data

In [13]:
```
# head
print(dataset.head(20))
```

```
    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0             6      148             72             35        0  33.6
1             1       85             66             29        0  26.6
2             8      183             64              0        0  23.3
3             1       89             66             23       94  28.1
4             0      137             40             35      168  43.1
5             5      116             74              0        0  25.6
6             3       78             50             32       88  31.0
7            10      115              0              0        0  35.3
8             2      197             70             45      543  30.5
9             8      125             96              0        0   0.0
10            4      110             92              0        0  37.6
11           10      168             74              0        0  38.0
12           10      139             80              0        0  27.1
13            1      189             60             23      846  30.1
14            5      166             72             19      175  25.8
15            7      100              0              0        0  30.0
16            0      118             84             47      230  45.8
17            7      107             74              0        0  29.6
18            1      103             30             38       83  43.3
19            1      115             70             30       96  34.6

    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
10                     0.191   30        0
11                     0.537   34        1
12                     1.441   57        0
13                     0.398   59        1
14                     0.587   51        1
15                     0.484   32        1
16                     0.551   31        1
17                     0.254   31        1
18                     0.183   33        0
19                     0.529   32        1
```

3.3 Statistical Summary

In [14]:
```
# descriptions
print(dataset.describe())
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

Following columns or variables have an invalid zero value: -Glucose -BloodPressure -SkinThickness -Insulin -BMI

It is better to replace zeros with nan since after that counting them would be easier and zeros need to be replaced with suitable values

In [40]:
```
diabetes_data_copy = dataset.copy(deep = True)
diabetes_data_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = diabetes_data_copy[['Glucose','BloodPressure','

## showing the count of Nans
print(diabetes_data_copy.isnull().sum())
diabetes_data_copy.head()
```

```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Out[40]:

	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

Aiming to impute nan values for the columns in accordance with their distribution

In [16]:
```
diabetes_data_copy['Glucose'].fillna(diabetes_data_copy['Glucose'].mean(), inplace = True)
diabetes_data_copy['BloodPressure'].fillna(diabetes_data_copy['BloodPressure'].mean(), inplace = True)
diabetes_data_copy['SkinThickness'].fillna(diabetes_data_copy['SkinThickness'].median(), inplace = True)
diabetes_data_copy['Insulin'].fillna(diabetes_data_copy['Insulin'].median(), inplace = True)
diabetes_data_copy['BMI'].fillna(diabetes_data_copy['BMI'].median(), inplace = True)
```

3.4 Outcome Distribution

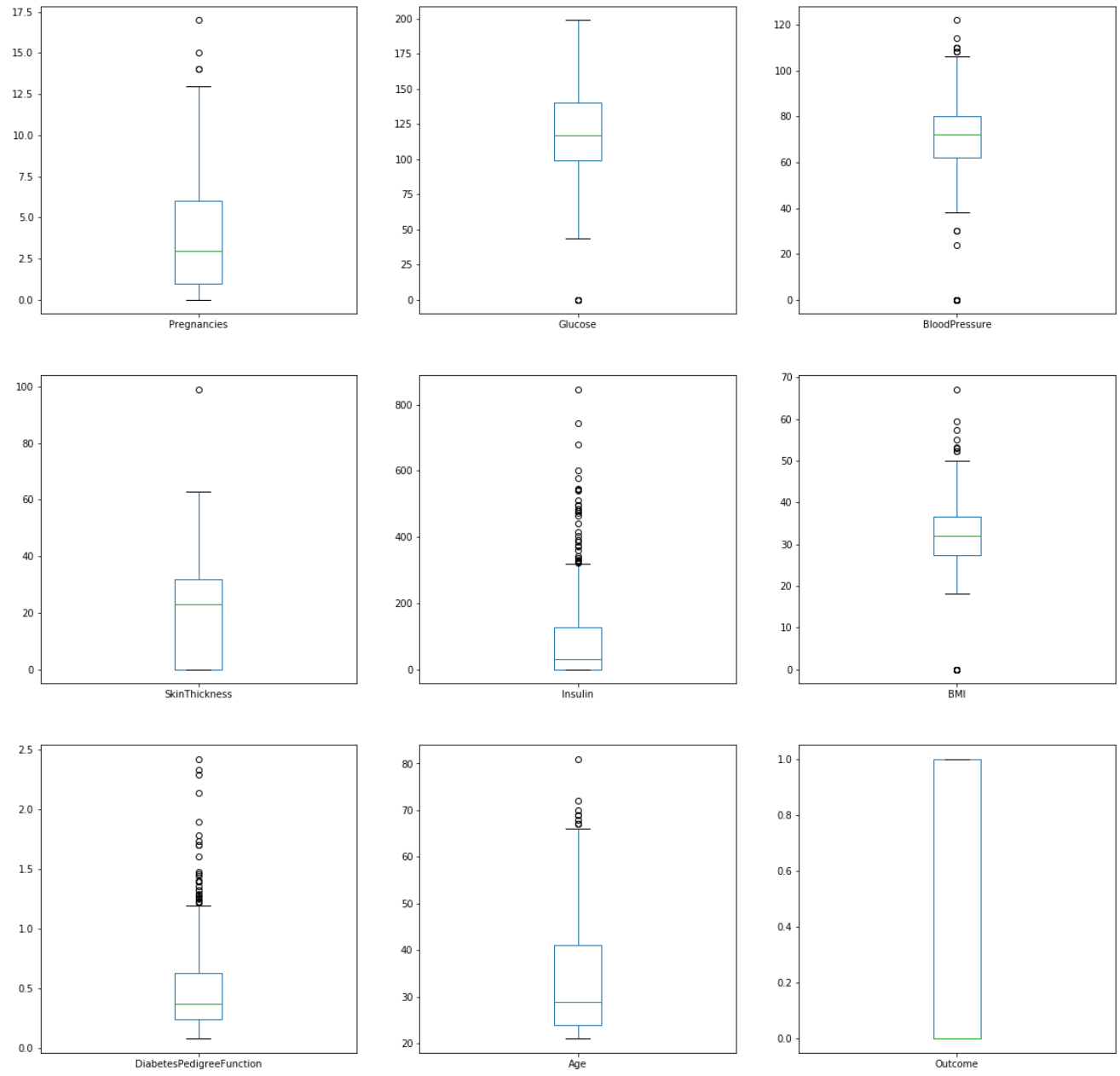
In [17]:
```
# outcome distribution
print(dataset.groupby('Outcome').size())
```

```
Outcome
0    500
1    268
dtype: int64
```

4. Data visualization

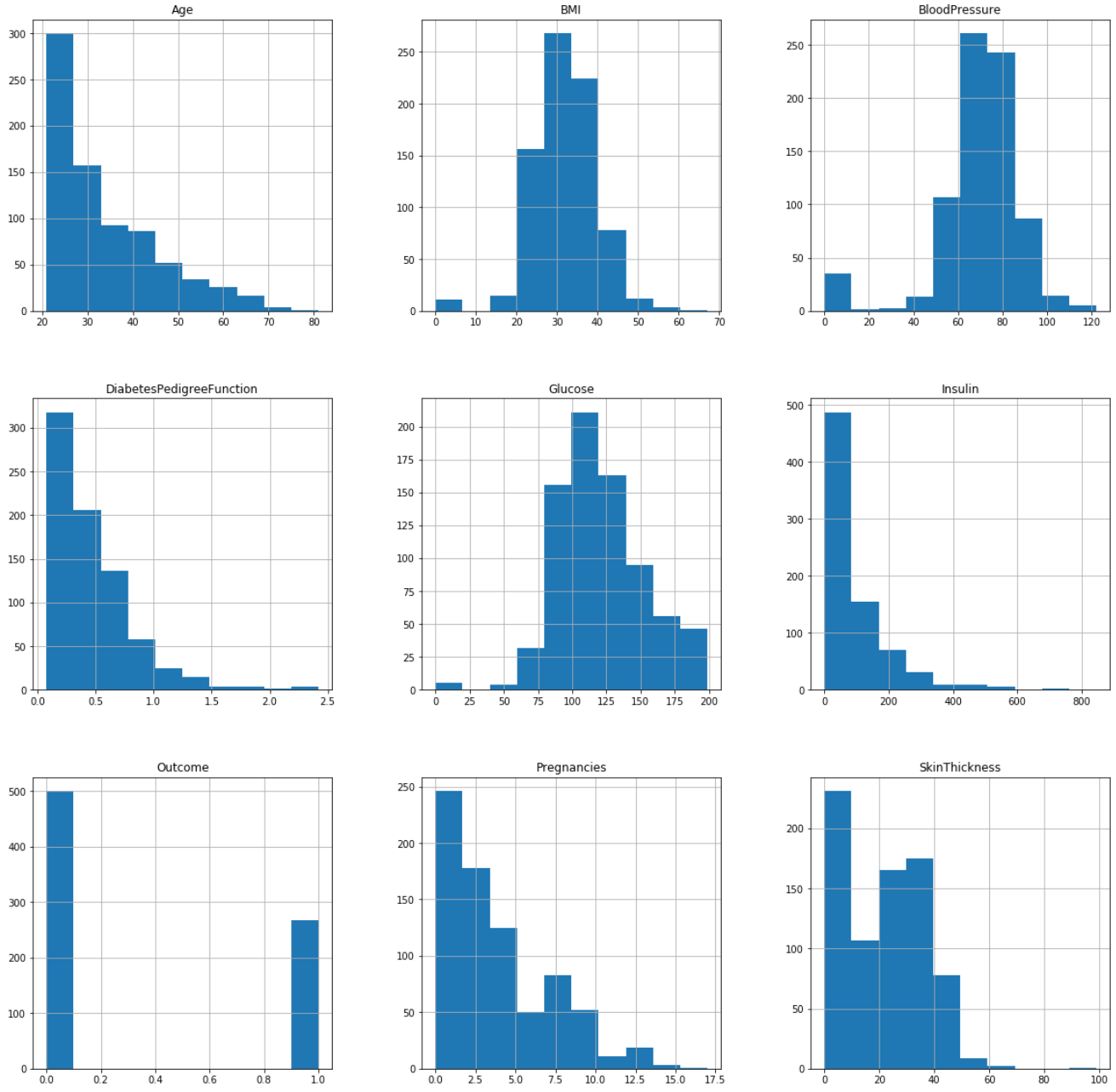
4.1 Univariate Plots

In [18]:
```
#adjust figure size first
pylab.rcParams['figure.figsize'] = (20, 20)
dataset.plot(kind='box', subplots=True,layout=(3,3))
plt.show()
```



we can also use histogram

In [19]:
```
#histograms
dataset.hist()
plt.show()
```

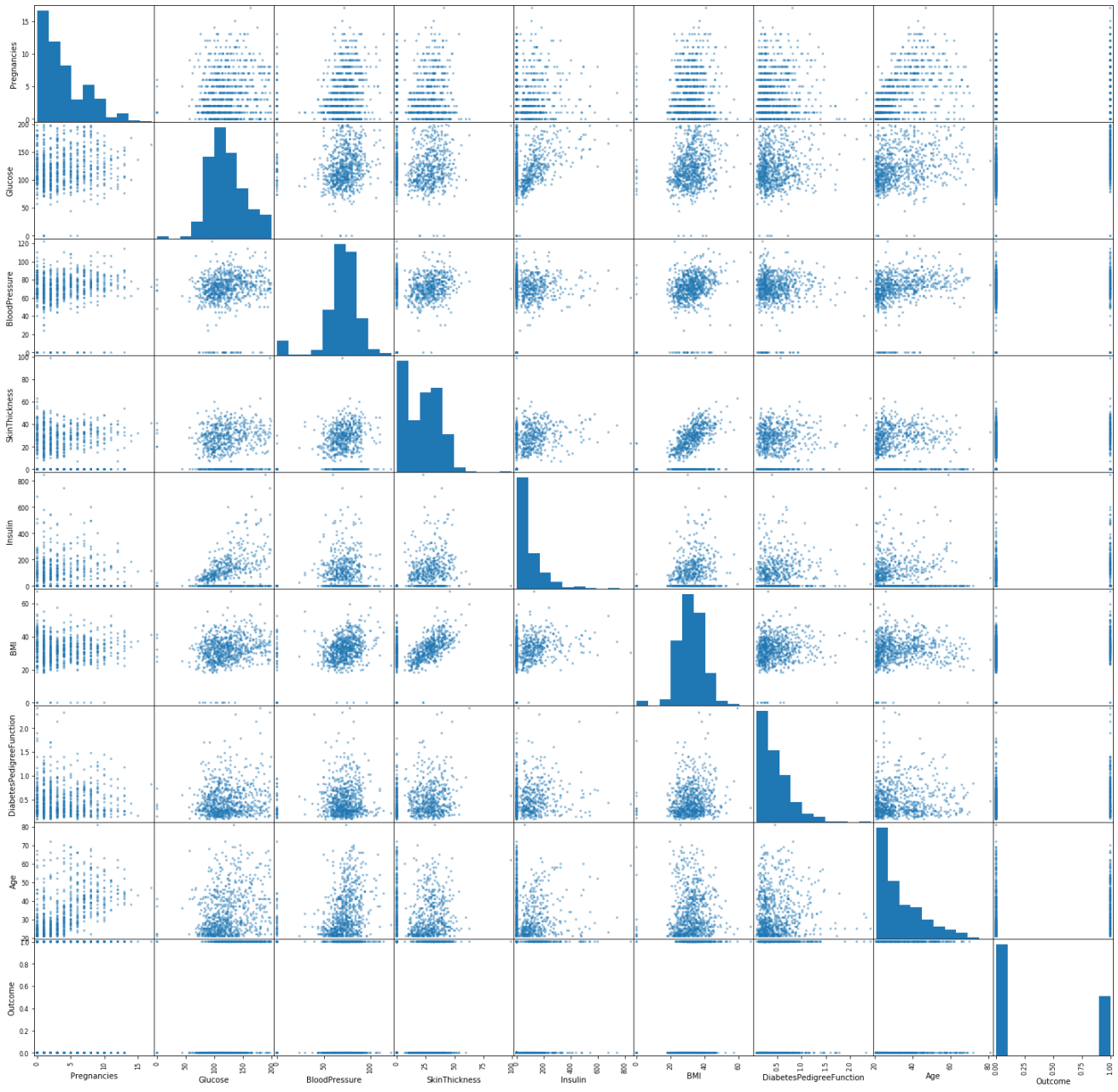


It looks like perhaps three of the input variables(BMI, BloodPressure and Glucose) have a Gaussian distribution. This is useful to note as we can use algorithms that can exploit this assumption.

4.2 Multivariate Plots

First, let's look at scatterplots of all pairs of attributes. This can be helpful to spot structured relationships between input variables.

In [20]:
```
# scatter plot matrix
pylab.rcParams['figure.figsize'] = (25, 25)
scatter_matrix(dataset)
plt.show()
```



Note the diagonal grouping of some pairs of attributes. This suggests a high correlation and a predictable relationship.

4.3 Pair plot for clean data

In [21]:
```
p=sns.pairplot(diabetes_data_copy, hue = 'Outcome')
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
<ipython-input-21-5abb6e74267f> in <module>()
----> 1 p=sns.pairplot(diabetes_data_copy, hue = 'Outcome')

/usr/local/lib/python3.5/dist-packages/seaborn/axisgrid.py in pairplot(data, hue, hue_order, palette, vars, x_vars, y_vars, kin
d, diag_kind, markers, height, aspect, dropna, plot_kws, diag_kws, grid_kws, size)
   2109             diag_kws.setdefault("shade", True)
   2110             diag_kws["legend"] = False
-> 2111             grid.map_diag(kdeplot, **diag_kws)
   2112
   2113     # Maybe plot on the off-diagonals

/usr/local/lib/python3.5/dist-packages/seaborn/axisgrid.py in map_diag(self, func, **kwargs)
   1397                     color = fixed_color
   1398
-> 1399                 func(data_k, label=label_k, color=color, **kwargs)
   1400
   1401             self._clean_axis(ax)

/usr/local/lib/python3.5/dist-packages/seaborn/distributions.py in kdeplot(data, data2, shade, vertical, kernel, bw, gridsize,
 cut, clip, legend, cumulative, shade_lowest, cbar, cbar_ax, cbar_kws, ax, **kwargs)
    689         ax = _univariate_kdeplot(data, shade, vertical, kernel, bw,
    690                                  gridsize, cut, clip, legend, ax,
--> 691                                  cumulative=cumulative, **kwargs)
    692
    693     return ax

/usr/local/lib/python3.5/dist-packages/seaborn/distributions.py in _univariate_kdeplot(data, shade, vertical, kernel, bw, grids
ize, cut, clip, legend, ax, cumulative, **kwargs)
    292                                "only implemented in statsmodels."
    293                                "Please install statsmodels.")
--> 294         x, y = _scipy_univariate_kde(data, bw, gridsize, cut, clip)
    295
    296     # Make sure the density is nonnegative

/usr/local/lib/python3.5/dist-packages/seaborn/distributions.py in _scipy_univariate_kde(data, bw, gridsize, cut, clip)
    364     """Compute a univariate kernel density estimate using scipy."""
    365     try:
--> 366         kde = stats.gaussian_kde(data, bw_method=bw)
    367     except TypeError:
    368         kde = stats.gaussian_kde(data)

/usr/local/lib/python3.5/dist-packages/scipy/stats/kde.py in __init__(self, dataset, bw_method)
    170
    171         self.d, self.n = self.dataset.shape
--> 172         self.set_bandwidth(bw_method=bw_method)
    173
    174     def evaluate(self, points):

/usr/local/lib/python3.5/dist-packages/scipy/stats/kde.py in set_bandwidth(self, bw_method)
    497             raise ValueError(msg)
    498
--> 499         self._compute_covariance()
    500
    501     def _compute_covariance(self):

/usr/local/lib/python3.5/dist-packages/scipy/stats/kde.py in _compute_covariance(self)
    508             self._data_covariance = atleast_2d(np.cov(self.dataset, rowvar=1,
    509                                                       bias=False))
--> 510             self._data_inv_cov = linalg.inv(self._data_covariance)
    511
    512         self.covariance = self._data_covariance * self.factor**2

/usr/local/lib/python3.5/dist-packages/scipy/linalg/basic.py in inv(a, overwrite_a, check_finite)
    817         inv_a, info = getri(lu, piv, lwork=lwork, overwrite_lu=1)
    818     if info > 0:
--> 819         raise LinAlgError("singular matrix")
    820     if info < 0:
    821         raise ValueError('illegal value in %d-th argument of internal '

LinAlgError: singular matrix
```

4.4 A heat map

heat map is a two-dimensional representation of information with the help of colors. Heat maps can help the user visualize simple or complex information.

In [22]:
```
plt.figure(figsize=(12,10))  # on this line I just set the size of figure to 12 by 10.
p=sns.heatmap(dataset.corr(), annot=True,cmap ='RdYlGn')  # seaborn has very simple solution for heatmap
```

4.5 Heatmap for clean data

In [23]:
```
plt.figure(figsize=(12,10))  # on this line I just set the size of figure to 12 by 10.
p=sns.heatmap(diabetes_data_copy.corr(), annot=True,cmap ='RdYlGn')  # seaborn has very simple solution for heatmap
```

4.6 Scaling the data to standard normal distribution

In [24]:
```
sc_X = StandardScaler()
X =  pandas.DataFrame(sc_X.fit_transform(diabetes_data_copy.drop(["Outcome"],axis = 1),),
        columns=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'])
```

In [25]:
```
X.head()
```

Out[25]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	0.639947	0.865108	-0.033518	0.670643	-0.181541	0.166619	0.468492	1.425995
1	-0.844885	-1.206162	-0.529859	-0.012301	-0.181541	-0.852200	-0.365061	-0.190672
2	1.233880	2.015813	-0.695306	-0.012301	-0.181541	-1.332500	0.604397	-0.105584
3	-0.844885	-1.074652	-0.529859	-0.695245	-0.540642	-0.633881	-0.920763	-1.041549
4	-1.141852	0.503458	-2.680669	0.670643	0.316566	1.549303	5.484909	-0.020496

In [46]:
```
#X = diabetes_data.drop("Outcome",axis = 1)
y = diabetes_data_copy.Outcome
```

5. Evaluate some algorithms

it is time to create some models

5.1 Create a Validation Dataset

split the loaded dataset into two, 80% of which we will use to train our models and 20% that we will hold back as a validation dataset.

In [49]:
```
# Split-out validation dataset

#X = array[:,0:4]
Y = diabetes_data_copy.Outcome
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=validation_size, random_state=see
```

We now have training data in the X_train and Y_train for preparing models and a X_validation and Y_validation sets that we can use later.

5.2 Test

We will use 10-fold cross validation to estimate accuracy. This will split our dataset into 10 parts, train on 9 and test on 1 and repeat for all combinations of train-test splits.

In [50]:
```
# Test options and evaluation metric
seed = 7
scoring = 'accuracy'
```

We are using the metric of ‘accuracy‘ to evaluate models. This is a ratio of the number of correctly predicted instances in divided by the total number of instances in the dataset multiplied by 100 to give a percentage (e.g. 95% accurate). We will be using the scoring variable when we run build and evaluate each model next.

5.3 Build Models

Let’s evaluate 2 different algorithms: -DTree -K-Nearest Neighbors (KNN).

In [52]:
```
# Spot Check Algorithms
models = []
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
KNN: 0.710233 (0.035825)
CART: 0.641777 (0.034062)
```

In [53]:
```
import autosklearn.classification
```

In [54]:
```
cls = autosklearn.classification.AutoSklearnClassifier()
```

In [55]:
```
cls.fit(X_train, Y_train)
```

```
[WARNING] [2019-07-27 23:04:48,631:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:04:48,631:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:16:12,607:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:16:12,607:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:38:40,536:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:38:40,536:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:39:18,342:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:39:18,342:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:40:24,116:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:40:24,116:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:41:07,950:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:41:07,950:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:42:19,968:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:42:19,968:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:44:06,309:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:44:06,309:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:46:57,096:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:46:57,096:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:48:01,925:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:48:01,925:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:59:56,513:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
[WARNING] [2019-07-27 23:59:56,513:smac.intensification.intensification.Intensifier] Challenger was the same as the current inc
umbent; Skipping challenger
```

In [56]:
```
predictions = cls.predict(X_validation)
```

In [57]:
```
import sklearn.metrics
print("Accuracy score", sklearn.metrics.accuracy_score(Y_validation, predictions))
```

```
Accuracy score 0.746753246753
```

In [ ]: