To predict the Diabetics Patients using PIMA Diabetics dataset

- 1 importing the libraries

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
import xgboost
```

In [3]:
```
data=pd.read_csv('~/Downloads/pima_data.csv')
```

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

Out[4]:

	num_preg	glucose_conc	diastolic_bp	thickness	insulin	bmi	diab_pred	age	skin	diabetes
0	6	148	72	35	0	33.6	0.627	50	1.3790	True
1	1	85	66	29	0	26.6	0.351	31	1.1426	False
2	8	183	64	0	0	23.3	0.672	32	0.0000	True
3	1	89	66	23	94	28.1	0.167	21	0.9062	False
4	0	137	40	35	168	43.1	2.288	33	1.3790	True

In [5]:
```
data.shape
```

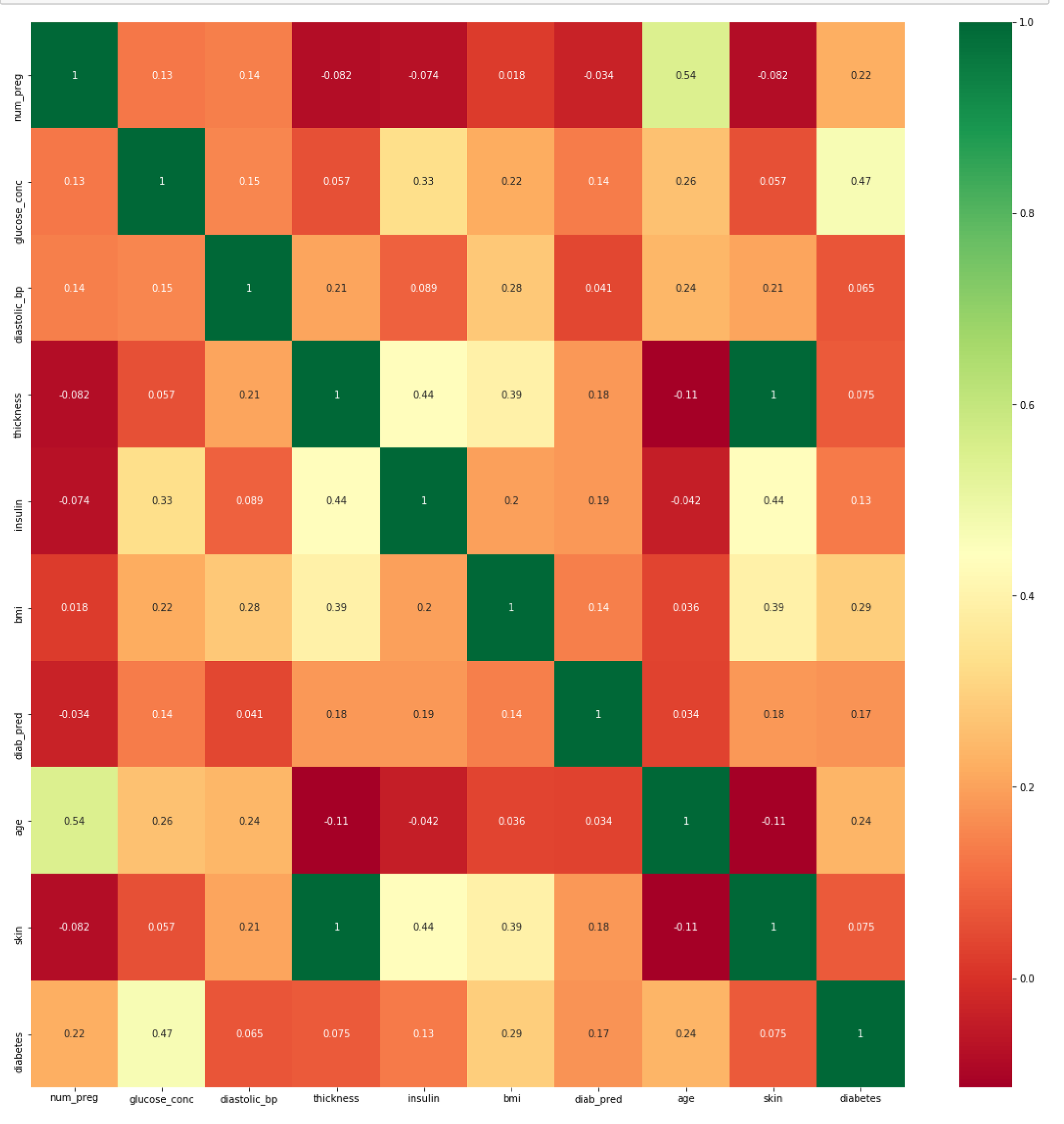
Out[5]:
```
(768, 10)
```

In [6]:
```
### checking if any null value is present
```

In [7]:
```
data.isnull().values.any()
```

Out[7]:
```
False
```

In [8]:
```
## corelation
import seaborn as sns
import matplotlib.pyplot as plt
#get correlations of each features in dataset
corrmat = data.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



In [9]:
```
data.corr()
```

Out[9]:

	num_preg	glucose_conc	diastolic_bp	thickness	insulin	bmi	diab_pred	age	skin	diabete
num_preg	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	-0.081672	0.22189
glucose_conc	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.057328	0.46658
diastolic_bp	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.207371	0.06506
thickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	1.000000	0.07475
insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.436783	0.13054
bmi	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.392573	0.29269
diab_pred	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.183928	0.17384
age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	-0.113970	0.23835
skin	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	1.000000	0.07475
diabetes	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	0.074752	1.00000

Changing the diabetes column data from boolean to number

In [10]:
```
diabetes_map = {True: 1, False: 0}
```

In [11]:
```
data['diabetes'] = data['diabetes'].map(diabetes_map)
```

In [12]:
```
data.head()
```

Out[12]:

	num_preg	glucose_conc	diastolic_bp	thickness	insulin	bmi	diab_pred	age	skin	diabetes
0	6	148	72	35	0	33.6	0.627	50	1.3790	1
1	1	85	66	29	0	26.6	0.351	31	1.1426	0
2	8	183	64	0	0	23.3	0.672	32	0.0000	1
3	1	89	66	23	94	28.1	0.167	21	0.9062	0
4	0	137	40	35	168	43.1	2.288	33	1.3790	1

In [13]:
```
diabetes_true_count = len(data.loc[data['diabetes'] == True])
diabetes_false_count = len(data.loc[data['diabetes'] == False])
```

In [14]:
```
(diabetes_true_count,diabetes_false_count)
```

Out[14]:
```
(268, 500)
```

In [15]:
```
## Train Test Split

from sklearn.model_selection import train_test_split
feature_columns = ['num_preg', 'glucose_conc', 'diastolic_bp', 'insulin', 'bmi', 'diab_pred'
, 'age', 'skin']
predicted_class = ['diabetes']
```

In [16]:
```
X = data[feature_columns].values
y = data[predicted_class].values


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state=10)
```

Check how many other missing(zero) values

In [17]:
```
print("total number of rows : {0}".format(len(data)))
print("number of rows missing glucose_conc: {0}".format(len(data.loc[data['glucose_conc'] ==
0])))
print("number of rows missing glucose_conc: {0}".format(len(data.loc[data['glucose_conc'] ==
0])))
print("number of rows missing diastolic_bp: {0}".format(len(data.loc[data['diastolic_bp'] ==
0])))
print("number of rows missing insulin: {0}".format(len(data.loc[data['insulin'] == 0])))
print("number of rows missing bmi: {0}".format(len(data.loc[data['bmi'] == 0])))
print("number of rows missing diab_pred: {0}".format(len(data.loc[data['diab_pred'] == 0])))
print("number of rows missing age: {0}".format(len(data.loc[data['age'] == 0])))
print("number of rows missing skin: {0}".format(len(data.loc[data['skin'] == 0])))
```

```
total number of rows : 768
number of rows missing glucose_conc: 5
number of rows missing glucose_conc: 5
number of rows missing diastolic_bp: 35
number of rows missing insulin: 374
number of rows missing bmi: 11
number of rows missing diab_pred: 0
number of rows missing age: 0
number of rows missing skin: 227
```

In [35]:
```
from sklearn.impute import SimpleImputer

fill_values = SimpleImputer(missing_values=0, strategy="mean")

X_train = fill_values.fit_transform(X_train)
X_test = fill_values.fit_transform(X_test)
```

In [37]:
```
X_train
```

Out[37]:
```
array([[2.0000e+00, 8.9000e+01, 9.0000e+01, ..., 2.9200e-01, 4.2000e+01,
        1.1820e+00],
       [4.0000e+00, 1.4600e+02, 8.5000e+01, ..., 1.8900e-01, 2.7000e+01,
        1.0638e+00],
       [1.0000e+01, 1.1100e+02, 7.0000e+01, ..., 1.4100e-01, 4.0000e+01,
        1.0638e+00],
       ...,
       [3.0000e+00, 1.1600e+02, 7.4000e+01, ..., 1.0700e-01, 2.4000e+01,
        5.9100e-01],
       [1.0000e+00, 8.8000e+01, 3.0000e+01, ..., 4.9600e-01, 2.6000e+01,
        1.6548e+00],
       [5.0000e+00, 9.6000e+01, 7.4000e+01, ..., 9.9700e-01, 4.3000e+01,
        7.0920e-01]])
```

In [38]:
```
X_test
```

Out[38]:
```
array([[  4.        , 154.        ,  72.        , ...,   0.338     ,
         37.        ,   1.1426    ],
       [  2.        , 112.        ,  86.        , ...,   0.246     ,
         28.        ,   1.6548    ],
       [  1.        , 135.        ,  54.        , ...,   0.687     ,
         62.        ,   1.17406711],
       ...,
       [  3.        , 150.        ,  76.        , ...,   0.207     ,
         37.        ,   1.17406711],
       [  3.        , 130.        ,  64.        , ...,   0.314     ,
         22.        ,   1.17406711],
       [  4.81578947, 108.        ,  68.        , ...,   0.787     ,
         32.        ,   0.788     ]])
```

In [39]:
```
## Apply Algorithm

from sklearn.ensemble import RandomForestClassifier
random_forest_model = RandomForestClassifier(random_state=10)

random_forest_model.fit(X_train, y_train.ravel())
```

Out[39]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=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=10, verbose=0,
                       warm_start=False)
```

In [ ]:

In [40]:
```
predict_train_data = random_forest_model.predict(X_test)

from sklearn import metrics

print("Accuracy = {0:.3f}".format(metrics.accuracy_score(y_test, predict_train_data)))
```

```
Accuracy = 0.736
```

In [41]:
```
## Hyper Parameter Optimization

params={
 "learning_rate"    : [0.05, 0.10, 0.15, 0.20, 0.25, 0.30 ] ,
 "max_depth"        : [ 3, 4, 5, 6, 8, 10, 12, 15],
 "min_child_weight" : [ 1, 3, 5, 7 ],
 "gamma"            : [ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
 "colsample_bytree" : [ 0.3, 0.4, 0.5 , 0.7 ]

}
```

In [42]:
```
## Hyperparameter optimization using RandomizedSearchCV
from sklearn.model_selection import RandomizedSearchCV
import xgboost
```

In [23]:
```
conda install -c anaconda py-xgboost
```

```
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /Users/abhilashavadhanula/Downloads/anaconda3

  added / updated specs:
    - py-xgboost


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    _py-xgboost-mutex-2.0      |            cpu_0           8 KB  anaconda
    ca-certificates-2020.1.1   |                0         132 KB  anaconda
    certifi-2019.11.28         |           py36_1         157 KB  anaconda
    conda-4.8.3                |           py36_0         3.0 MB  anaconda
    libxgboost-0.90            |       h0a44026_1         2.4 MB  anaconda
    py-xgboost-0.90            |   py36h0a44026_1          76 KB  anaconda
    ------------------------------------------------------------
                                           Total:         5.8 MB

The following NEW packages will be INSTALLED:

  _py-xgboost-mutex  anaconda/osx-64::_py-xgboost-mutex-2.0-cpu_0
  libxgboost         anaconda/osx-64::libxgboost-0.90-h0a44026_1
  py-xgboost         anaconda/osx-64::py-xgboost-0.90-py36h0a44026_1

The following packages will be UPDATED:

  certifi              pkgs/main::certifi-2019.11.28-py36_0 --> anaconda::certifi-2019.11.28-
py36_1
  openssl              pkgs/main::openssl-1.1.1e-h1de35cc_0 --> anaconda::openssl-1.1.1-h1de3
5cc_0

The following packages will be SUPERSEDED by a higher-priority channel:

  ca-certificates                                 pkgs/main --> anaconda
  conda                                           pkgs/main --> anaconda



Downloading and Extracting Packages
py-xgboost-0.90      | 76 KB     | ##################################### | 100%
ca-certificates-2020 | 132 KB    | ##################################### | 100%
libxgboost-0.90      | 2.4 MB    | ##################################### | 100%
certifi-2019.11.28   | 157 KB    | ##################################### | 100%
conda-4.8.3          | 3.0 MB    | ##################################### | 100%
_py-xgboost-mutex-2. | 8 KB      | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [43]:
```
classifier=xgboost.XGBClassifier()
random_search=RandomizedSearchCV(classifier,param_distributions=params,n_iter=5,scoring='roc
_auc',n_jobs=-1,cv=5,verbose=3)
```

In [44]:
```
def timer(start_time=None):
    if not start_time:
        start_time = datetime.now()
        return start_time
    elif start_time:
        thour, temp_sec = divmod((datetime.now() - start_time).total_seconds(), 3600)
        tmin, tsec = divmod(temp_sec, 60)
        print('\n Time taken: %i hours %i minutes and %s seconds.' % (thour, tmin, round(tse
c, 2)))
```

In [46]:
```
from datetime import datetime
# Here we go
start_time = timer(None) # timing starts from this point for "start_time" variable
random_search.fit(X,y.ravel())
timer(start_time) # timing ends here for "start_time" variable
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.

Fitting 5 folds for each of 5 candidates, totalling 25 fits

[Parallel(n_jobs=-1)]: Done  18 out of  25 | elapsed:    1.0s remaining:    0.4s

 Time taken: 0 hours 0 minutes and 1.42 seconds.

[Parallel(n_jobs=-1)]: Done  25 out of  25 | elapsed:    1.3s finished
```

In [47]:
```
random_search.best_estimator_
```

Out[47]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.4, gamma=0.4,
              learning_rate=0.05, max_delta_step=0, max_depth=15,
              min_child_weight=3, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

In [48]:
```
classifier=xgboost.XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=0.3, gamma=0.0, learning_rate=0.25,
       max_delta_step=0, max_depth=3, min_child_weight=7, missing=None,
       n_estimators=100, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=True,
       subsample=1)
```

In [49]:
```
from sklearn.model_selection import cross_val_score
score=cross_val_score(classifier,X,y.ravel(),cv=10)
```

In [50]:
```
score
```

Out[50]:
```
array([0.67532468, 0.76623377, 0.74025974, 0.7012987 , 0.71428571,
       0.74025974, 0.76623377, 0.79220779, 0.77631579, 0.80263158])
```

In [51]:
```
score.mean()
```

Out[51]:
```
0.7475051264524948
```

In [ ]: