Abhishek Bedi

T00595450

Jatin Pant

T00608600

Rajat Mahajan

T00601202

Piper Jackson

COMP 4980_03

Machine Learning

April 13, 2021

The Analysis of Cause of Diabetes Among Females

Data Description

Pima Indians Diabetes Dataset

	A	B	C	D	E	F	G	H	I
1	Pregnancie	Glucose	BloodPres	SkinThickn	Insulin	BMI	DiabetesP	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1
9	10	115	0	0	0	35.3	0.134	29	0
10	2	197	70	45	543	30.5	0.158	53	1
11	8	125	96	0	0	0	0.232	54	1
12	4	110	92	0	0	37.6	0.191	30	0
13	10	168	74	0	0	38	0.537	34	1
14	10	139	80	0	0	27.1	1.441	57	0
15	1	189	60	23	846	30.1	0.398	59	1
16	5	166	72	19	175	25.8	0.587	51	1
17	7	100	0	0	0	30	0.484	32	1
18	0	118	84	47	230	45.8	0.551	31	1
19	7	107	74	0	0	29.6	0.254	31	1
20	1	103	30	38	83	43.3	0.183	33	0
21	1	115	70	30	96	34.6	0.529	32	1
22	3	126	88	41	235	39.3	0.704	27	0
23	8	99	84	0	0	35.4	0.388	50	0
24	7	196	90	0	0	39.8	0.451	41	1

It is a free to use dataset on kaggle.com. It has factors which cause diabetes as attributes and one outcome attribute. This dataset has 9 columns and 769 rows and the file size is 24 KB. There are some attributes with 0 values (excluding outcome and pregnancy column), these are missing values in this dataset which are replaced with 0. These values can be dropped as we cannot make up experimental values. After preprocessing, this dataset is really reliable. In this modern world, this dataset can be extremely useful as many nations are trying to fight obesity and diabetes, and this dataset can give a huge insight to these problems, which are usually overlooked. This dataset can help make predictions on cause of diabetes for a particular person and help prevent it. This dataset was last updated in 2016.

Data Analysis

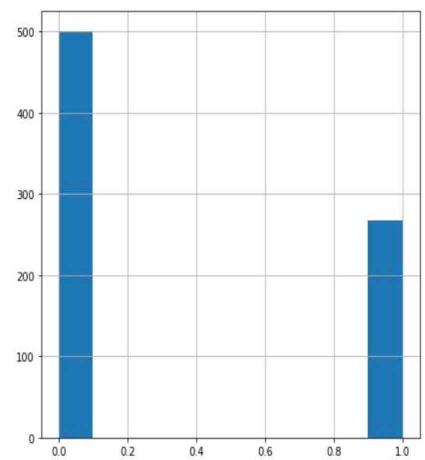
In this section, we are going thru every attribute of the dataset and try to find any correlations among them. We also created a diabetes_per_attribute function, which gives us a visualization of attribute and outcome together.

Attributes

Outcome

```
In [5]:  data['Outcome'].describe()

Out[5]:  count    768.000000
         mean       0.348958
         std        0.476951
         min        0.000000
         25%        0.000000
         50%        0.000000
         75%        1.000000
         max        1.000000
         Name: Outcome, dtype: float64
```

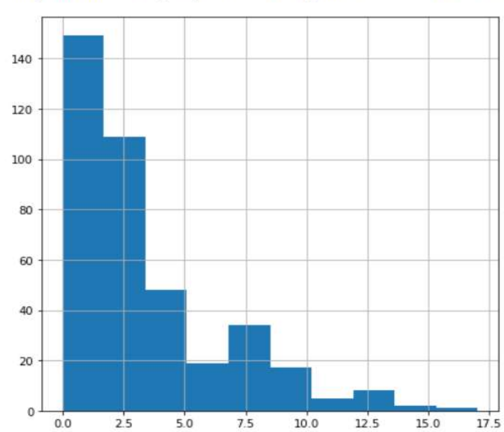
```
In [6]:  data['Outcome'].hist(figsize=(7,7))

Out[6]:  <matplotlib.axes._subplots.AxesSubplot at 0x20d2760ac88>
```



Pregnancies

```
In [10]:  data['Pregnancies'].describe()

Out[10]:  count    392.000000
          mean       3.301020
          std        3.211424
          min        0.000000
          25%        1.000000
          50%        2.000000
          75%        5.000000
          max       17.000000
          Name: Pregnancies, dtype: float64
```

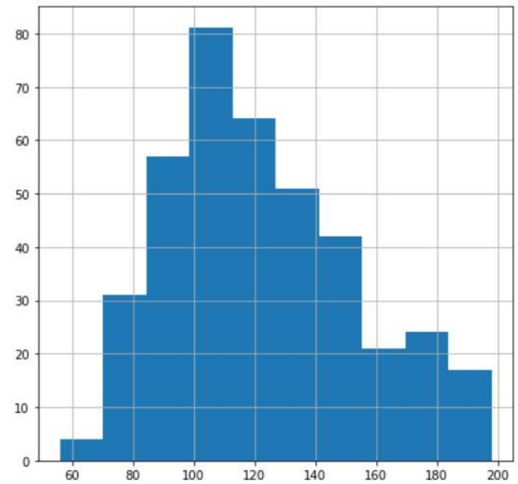
```
In [11]:  data['Pregnancies'].hist(figsize=(7,7))

Out[11]:  <matplotlib.axes._subplots.AxesSubplot at 0x20d27e15788>
```



Glucose

```
In [14]: data['Glucose'].describe()

Out[14]: count    392.000000
         mean     122.627551
         std       30.860781
         min       56.000000
         25%       99.000000
         50%      119.000000
         75%      143.000000
         max      198.000000
         Name: Glucose, dtype: float64
```

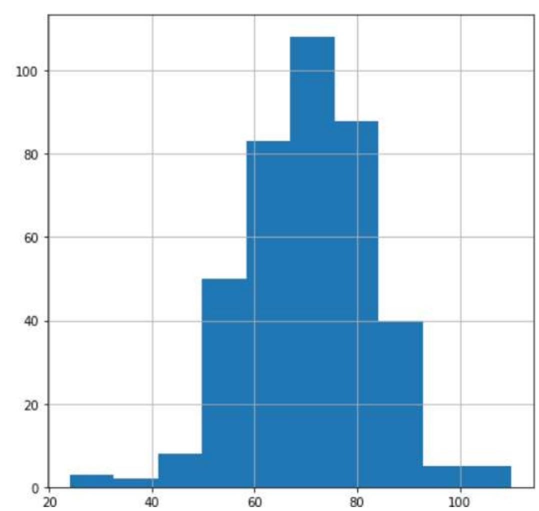
```
In [15]: data['Glucose'].hist(figsize=(7,7))

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x20d27f43908>
```



Blood Pressure

```
In [17]: data['BloodPressure'].hist(figsize=(7,7))

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x20d27f86908>
```

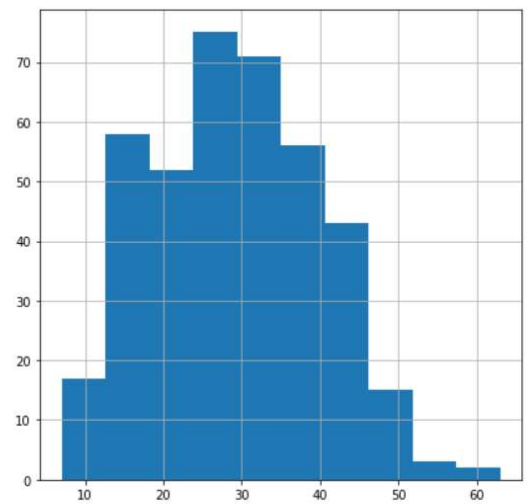
```
In [16]: data['BloodPressure'].describe()

Out[16]: count    392.000000
         mean      70.663265
         std       12.496092
         min       24.000000
         25%       62.000000
         50%       70.000000
         75%       78.000000
         max      110.000000
         Name: BloodPressure, dtype: float64
```



Skin Thickness

```
In [18]: data['SkinThickness'].describe()

Out[18]: count    392.000000
         mean      29.145408
         std       10.516424
         min        7.000000
         25%       21.000000
         50%       29.000000
         75%       37.000000
         max       63.000000
         Name: SkinThickness, dtype: float64
```

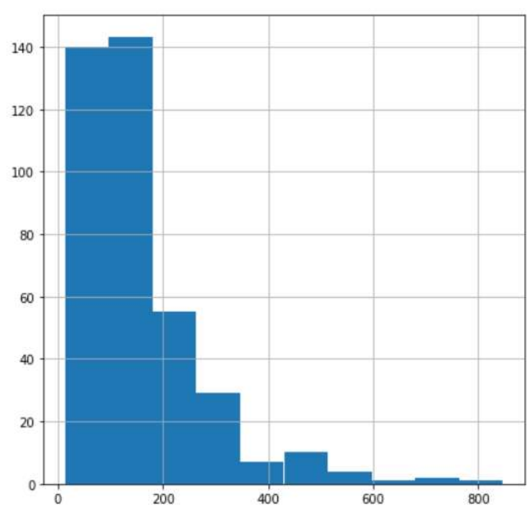
```
In [20]: data['SkinThickness'].hist(figsize=(7,7))
   Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x20d29059888>
```



Insulin

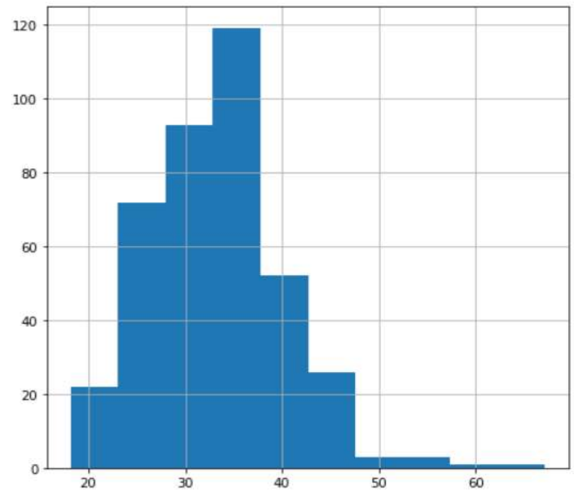
```
In [21]: data['Insulin'].describe()

Out[21]: count    392.000000
         mean     156.056122
         std      118.841690
         min       14.000000
         25%       76.750000
         50%      125.500000
         75%      190.000000
         max      846.000000
         Name: Insulin, dtype: float64
```

```
In [22]: data['Insulin'].hist(figsize=(7,7))
   Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x20d29126ec8>
```



BMI

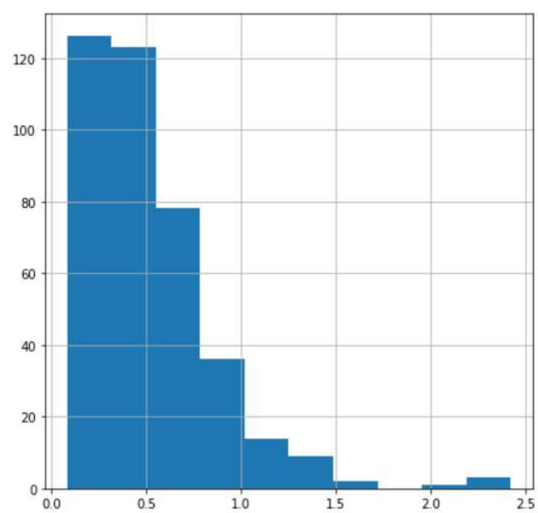
```
In [24]: data['BMI'].hist(figsize=(7,7))
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x20d291b5488>
```




```
In [23]: data['BMI'].describe()
Out[23]: count    392.000000
         mean      33.086224
         std        7.027659
         min       18.200000
         25%       28.400000
         50%       33.200000
         75%       37.100000
         max       67.100000
         Name: BMI, dtype: float64
```

Diabetes Pedigree Function

```
In [26]: data['DiabetesPedigreeFunction'].hist(figsize=(7,7))
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x20d29239908>
```

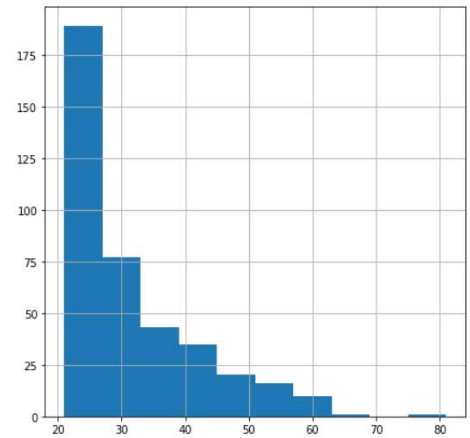



```
In [25]: data['DiabetesPedigreeFunction'].describe()
Out[25]: count    392.000000
         mean       0.523046
         std        0.345488
         min        0.085000
         25%        0.269750
         50%        0.449500
         75%        0.687000
         max        2.420000
         Name: DiabetesPedigreeFunction, dtype: float64
```

Age

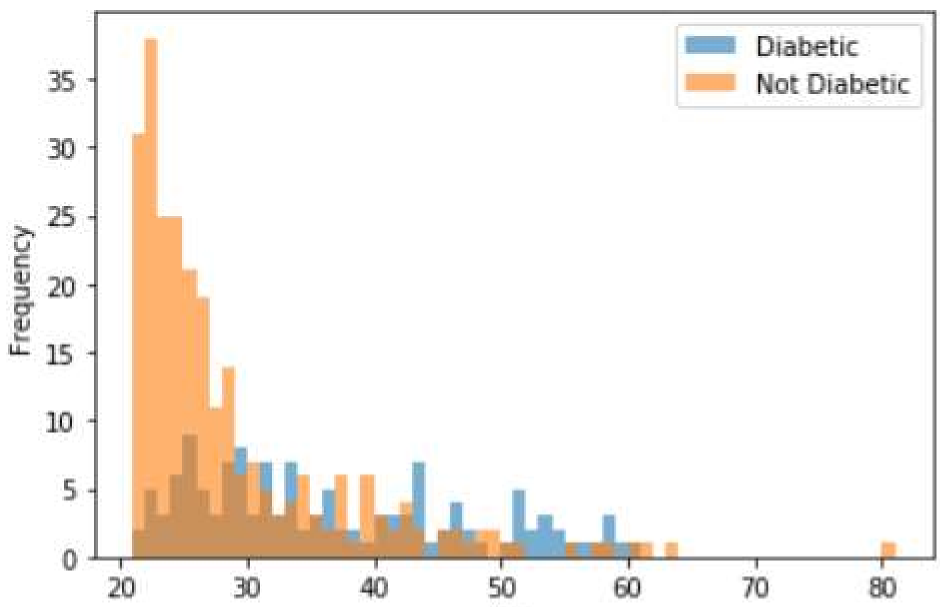
```
In [27]: data['Age'].describe()

Out[27]: count    392.000000
         mean      30.864796
         std       10.200777
         min       21.000000
         25%       23.000000
         50%       27.000000
         75%       36.000000
         max       81.000000
         Name: Age, dtype: float64
```

```
In [28]: data['Age'].hist(figsize=(7,7))

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x20d291b5588>
```



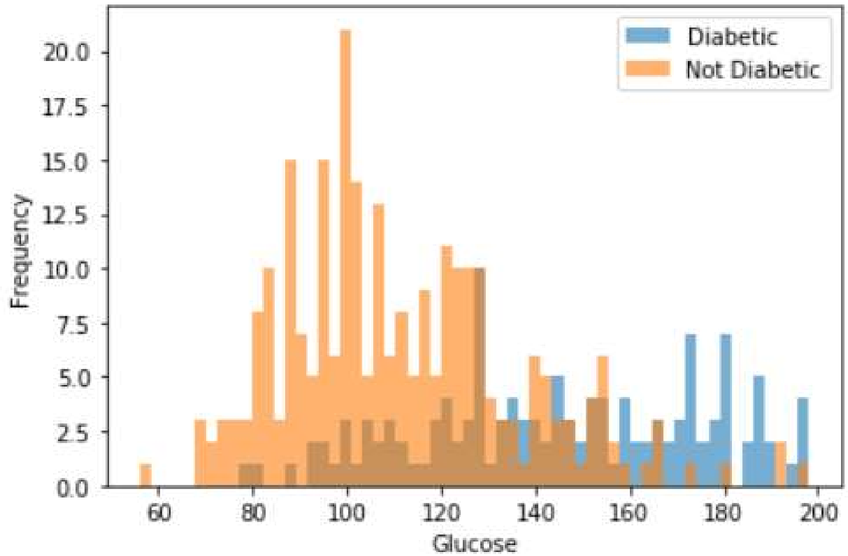
After this, we ran diabetic_per_attribute function for all attributes, some of which are below:

```
In [30]: plot_diabetic_per_attribute(data, "Age")
```



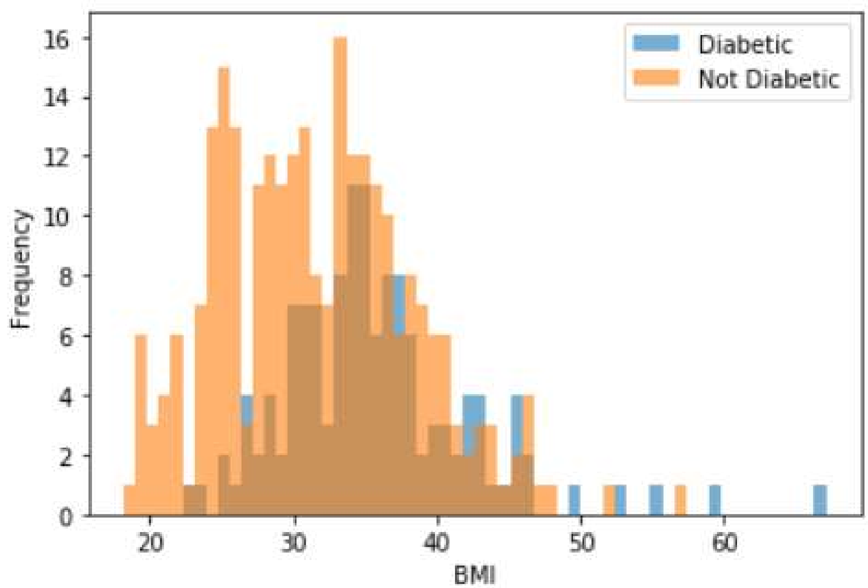
This shows us that as the older the person, more chance to be diabetic.

```
In [31]:  plot_diabetic_per_attribute(data, "Glucose")
```



This shows that more glucose means more change to get diabetes.

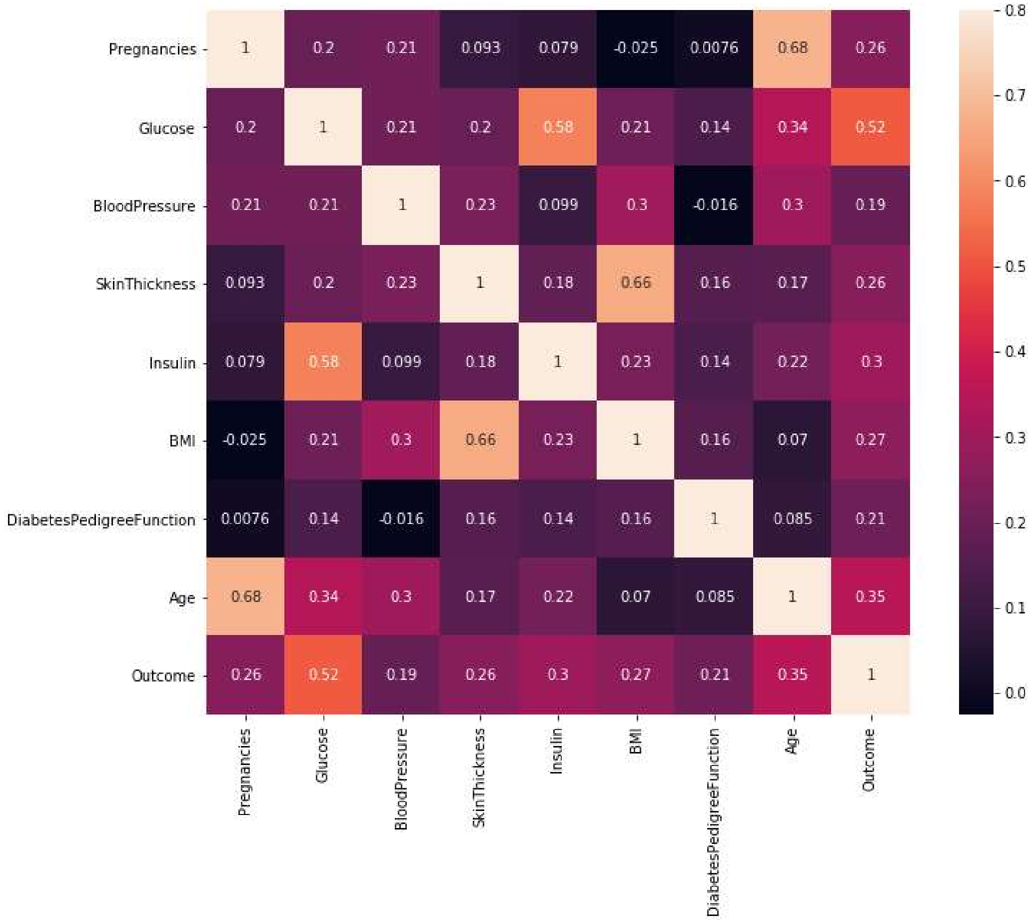
```
In [32]:  plot_diabetic_per_attribute(data, "BMI")
```



This gives us a relation between obesity and diabetes, more obese people have more chances of diabetes.

Correlation Matrix

In [33]:
```
import seaborn as sns

corrmat = data.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, cbar=True, annot=True, square=True, vmax=.8);
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.2	0.21	0.093	0.079	-0.025	0.0076	0.68	0.26
Glucose	0.2	1	0.21	0.2	0.58	0.21	0.14	0.34	0.52
BloodPressure	0.21	0.21	1	0.23	0.099	0.3	-0.016	0.3	0.19
SkinThickness	0.093	0.2	0.23	1	0.18	0.66	0.16	0.17	0.26
Insulin	0.079	0.58	0.099	0.18	1	0.23	0.14	0.22	0.3
BMI	-0.025	0.21	0.3	0.66	0.23	1	0.16	0.07	0.27
DiabetesPedigreeFunction	0.0076	0.14	-0.016	0.16	0.14	0.16	1	0.085	0.21
Age	0.68	0.34	0.3	0.17	0.22	0.07	0.085	1	0.35
Outcome	0.26	0.52	0.19	0.26	0.3	0.27	0.21	0.35	1

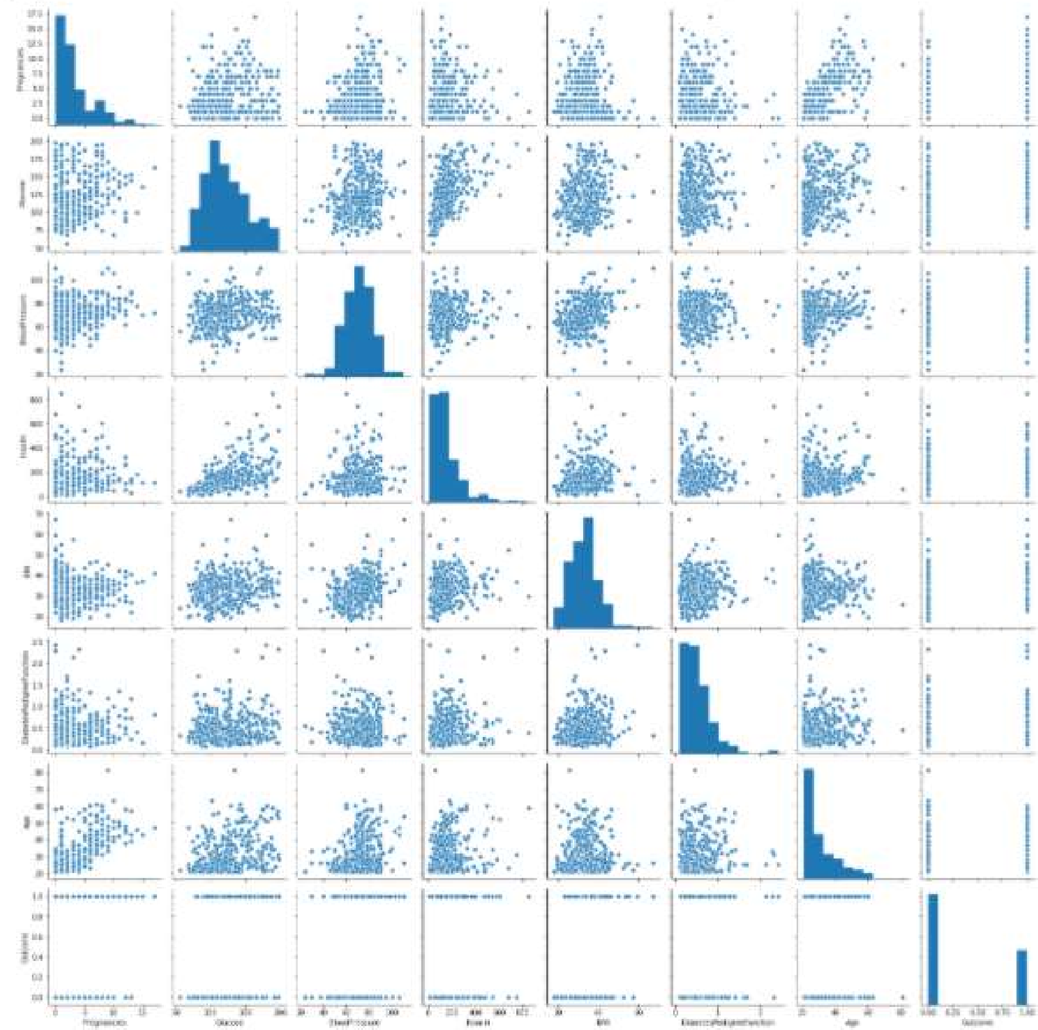
To break this matrix down:

- Glucose, Age and BMI are the most Correlated features with the 'Outcome'
- Bloodpressure, SkinThickness have tiny Correlation with the outcome.
- Age with Pregnancies are the most Correlated features.
- Insulin is correlated with Glucuse, which is a biological fact.
- DiabetesPedigreeFunction bit Correlated with most of them as it is calculated with taking all other attributes as inputs.

Scatter Plots

In [34]:

```
cols = ['Pregnancies','Glucose','BloodPressure','Insulin','BMI','DiabetesPedi
sns.pairplot(data[cols], size = 2.5)
plt.show();
```



Here, we can see there is an obvious relation between blood pressure and age, it is also obvious as blood pressure increases with age.

Data Exploration

Principle Component Analysis (PCA)

We used classification model, first we ran it with all the attributes and then did some more runs for less attributes.

Accuracy without PCA: 72.033%

Accuracy with PCA (all dimensions): 70.338%

Accuracy with PCA (7 dimensions, 99% variance): 73.728%

Accuracy with PCA (7 dimensions, 96% variance): 77.118%

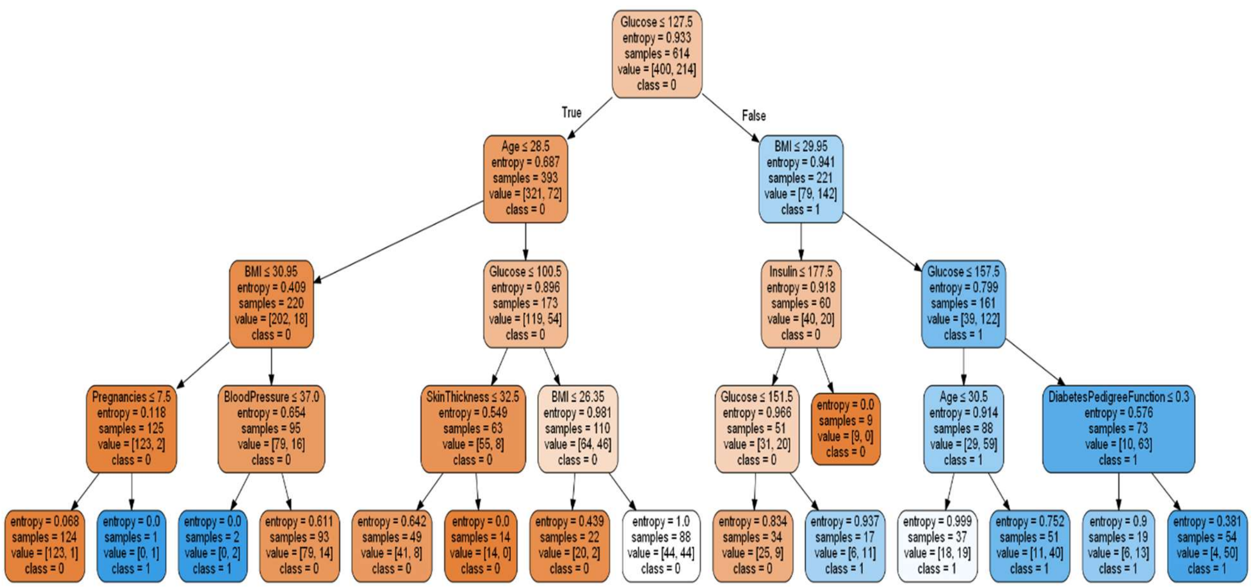
Accuracy with PCA (6 dimensions, 90% variance): 68.644%

It was clear after running code several time with PCA, that accuracy goes up when PCA with 3 or 2 dimensions is used. Accuracy was highest at 96% variance and all the 7 dimensions were necessary. As the dimensions kept decreasing, the accuracy also went down. By doing PCA analysis, we found that to get highest accuracy we need 7 dimensions, i.e., we need 7 attributes.

Decision Tree Algorithm

We used classification method for our dataset to create a decision tree and got a accuracy of 76.19%.

```
Accuracy: 0.7619047619047619
[[127  25]
 [ 30  49]]
```



It is clear from decision tree that Glucose levels <= 127.5 have only two cases for a woman to be diabetic. There is lots of case where woman can be diabetic if their glucose levels are more than 127.5. It is evident from decision tree that increase in glucose level is the main cause of diabetes in India. Also, if your BMI is less than 30, i.e you are not in Obese category, then there is less instances to be diabetic. On increasing the depth of decision tree much more information can be found.

Multi-Layered Perceptron & Random Forests

We used k-fold cross validation both of the algorithms to get good results.

```
In [186]:
cv = KFold(n_splits=10, random_state=1, shuffle=True)

model=MLPClassifier(max_iter=30000).fit(X_train,y_train)
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))

Accuracy: 0.776 (0.060)
```

After putting our dataset in MLP Classifier we got an accuracy of 77.6% which is almost what we go with PCA (96% variance) and it is also close to our decision tree algorithm.

```
In [191]:
model=RandomForestClassifier().fit(X_train,y_train)
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)
print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))

Accuracy: 0.796 (0.055)
```

After running for random forest classifier we got an accuracy of 79.6% which is the highest accuracy we got from all the models.

Result

We evaluated our algorithms based on accuracy.

Accuracy function

Using metrics library our accuracy for decision tree came out to be 0.77, which is 77% and is pretty good.

Confusion Matrix

Our confusion matrix was:

```
Accuracy: 0.7619047619047619
[[127  25]
 [ 30  49]]
```

This means my True Positives (TP) were 127

True Negatives (TN) were 25

False Positives (FP) were 30

False Negatives (FN) were 49

Accuracy according to results from confusion matrix is

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy = (127 + 25)/(127 + 25 + 30 + 49) = 152/231 = 0.768

Which is same as accuracy calculated by metrics, accuracy_score function.

The highest accuracy we got was with random forest classifier with cross validation which was 79.6%

Our machine learning model can be used in hospitals, where it can collect more data and improve, to predict and help people to lower the chances of diabetes. For example, we can collect data of an individual and let them know if their glucose levels go, above 40 they have high chance of getting diabetes so they should limit their sugar intake.

References

A. H. Fischl, “Classifications for Diabetes in Older Adults,” EndocrineWeb, 15-Apr-2016.

[Online]. Available: https://www.endocrineweb.com/guides/diabetes-older-people/risk-rises-age. [Accessed: 09-Apr-2021].

S. A. Kaveeshwar and J. Cornwall, “The current state of diabetes mellitus in India,” The Australasian medical journal, 31-Jan-2014. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3920109/. [Accessed: 10-Apr-2021].

F. Ceballos, “Scikit-Learn Decision Trees Explained,” Medium, 06-Apr-2021. [Online]. Available: https://towardsdatascience.com/scikit-learn-decision-trees-explained-803f3812290d. [Accessed: 07-Apr-2021].

A. Long, “Understanding Data Science Classification Metrics in Scikit-Learn in Python,” Medium, 09-Feb-2019. [Online]. Available: https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019. [Accessed: 04-Apr-2021].

“Confusion Matrix in Machine Learning,” GeeksforGeeks, 23-Feb-2021. [Online]. Available: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/. [Accessed: 09-Apr-2021].