Diabetes Veri Seti

Bu veri seti 768 hastaya ait 9 farklı değişkenden oluşmaktadır.

Değişkenler :

- Pregnancies
- Glucose
- BloodPressure
- SkinThickness
- Insulin
- BMI
- DiabetesPedigreeFunction
- Age
- Outcome

In [1]:
```
# Kütüphaneleri import edelim
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Python'da uyarıları kapatalım
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
# Veriyi içeri aktaralım
df = pd.read_csv("diabetes.csv")
df.head()
```

Out[2]:

	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 [3]:
```
# Veri hakkında bilgi edinelim
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [4]:
```
# Veri setinindeki istatistiksel değerlere bakalım
df.describe().T
```

Out[4]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00

Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

In [5]:
```
# Veri setinde boş değer var mı kontrol edelim
df.isnull().values.any()
```

Out[5]:
```
False
```

isnull komutu ile boş değer gözükmüyor ama insülin değerlerinde '0' görüyoruz. İnsülin değerleri '0' olamaz aynı zamanda '0'olmaması gereken başka değerlede mevcut fakat *outcome* değeri '0' içerebilir çünkü oradaki '0' ın anlamı negatif. Şimdi veri setindeki '0'lara bakalım ve sonra o değerleri 'NaN' ile dolduralım.

In [6]:
```
df.eq(0).sum()
```

Out[6]:
```
Pregnancies                 111
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

Eksik değer olan sıfıların bulunduğu sütunları getirelim ve '0' yerine 'NaN' yazalım.

In [7]:
```
df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI',
    'DiabetesPedigreeFunction','Age']]= df[['Glucose','BloodPressure',
    'SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']].replace(0,np.NaN)
```

Verisetimizde eksik olan değerlei '0' dan 'NaN'a çevirdik ama bu şekilde bırakmamalıyız eksik değerleri doldurmamız gerekiyor. Eksik değereleri ortalama değerler ile dolduralım.

In [8]:
```
# Eksik değerlerin doldurulması
df.fillna(df.mean(), inplace=True)
```

In [9]:
```
# Veri setimizin son haline bakalım
df.head()
```

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.00000	155.548223	33.6	0.627	50	1
1	1	85.0	66.0	29.00000	155.548223	26.6	0.351	31	0
2	8	183.0	64.0	29.15342	155.548223	23.3	0.672	32	1
3	1	89.0	66.0	23.00000	94.000000	28.1	0.167	21	0
4	0	137.0	40.0	35.00000	168.000000	43.1	2.288	33	1

In [10]:
```
# Eksik değerleri tekrar kontrol edelim
df.isnull().sum()
```

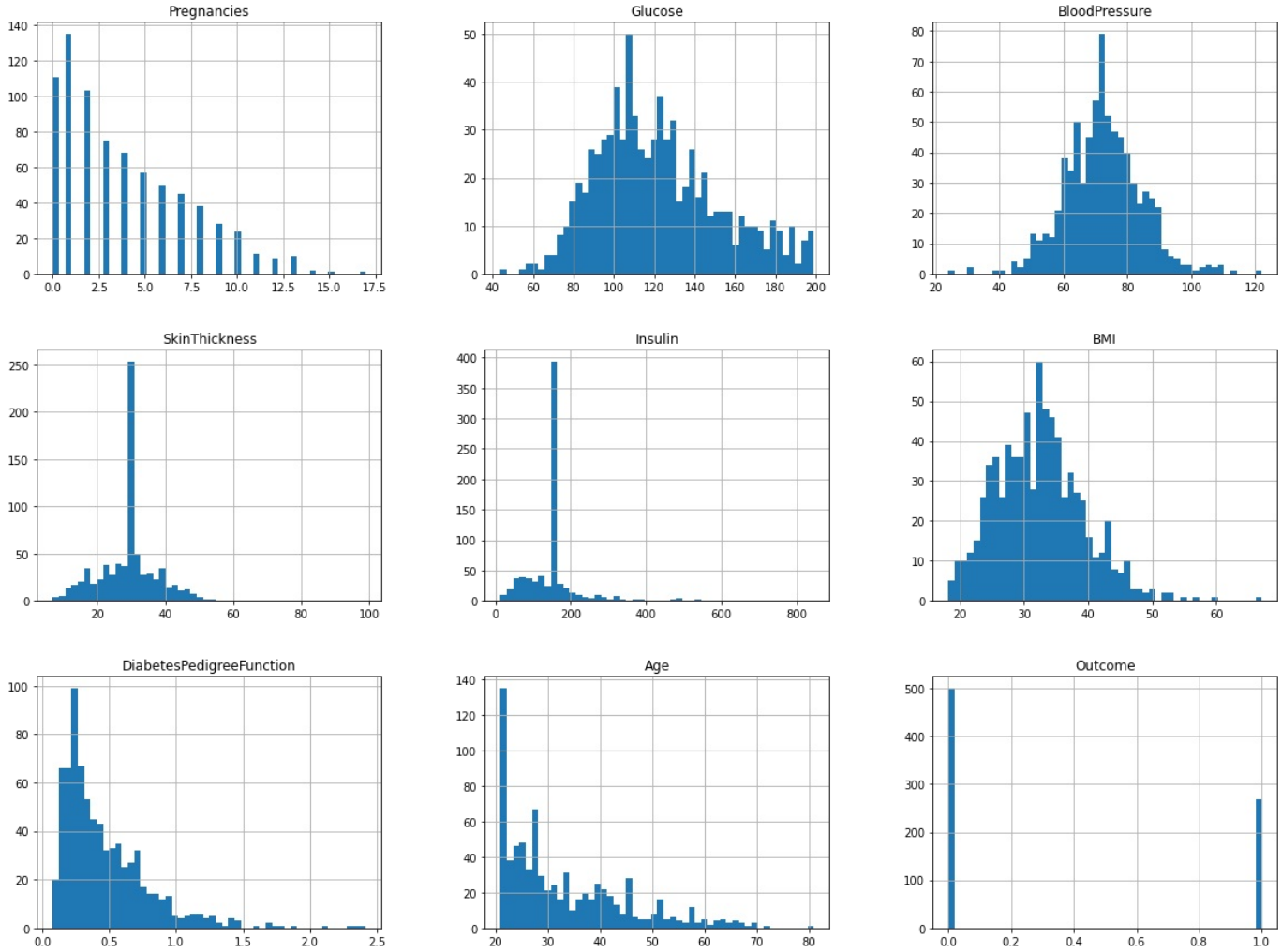
Out[10]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [11]:
```
# Veri setindeki 0 ları kontrol edelim
df.eq(0).sum()
```

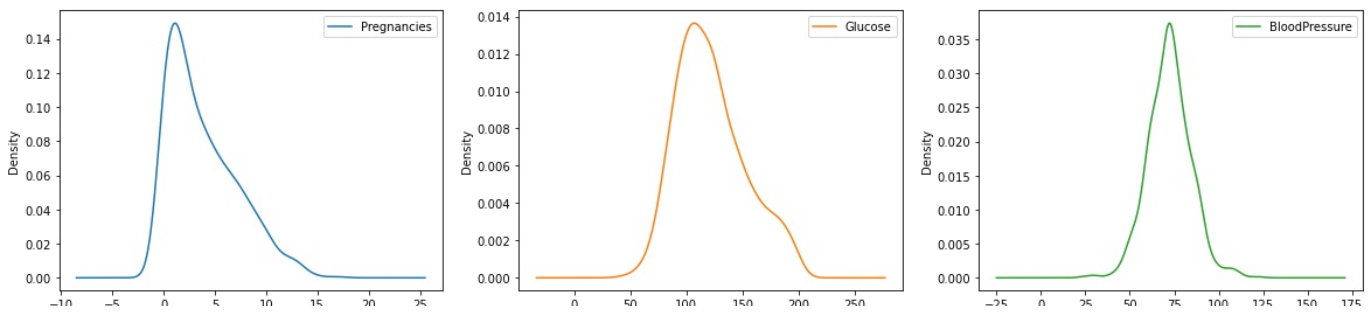
Out[11]:
```
Pregnancies                 111
Glucose                       0
BloodPressure                 0
SkinThickness                 0
Insulin                       0
BMI                           0
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

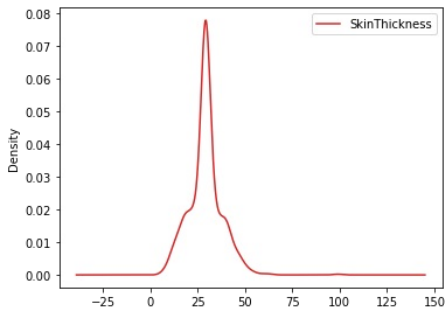
İstatistiksel Değerleri Kullanarak Grafikler Çizdirelim¶

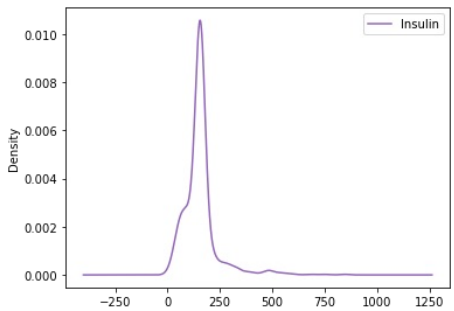
In [12]:
```
# Histogram
df.hist(bins = 50, figsize = (20,15))
plt.show()
```

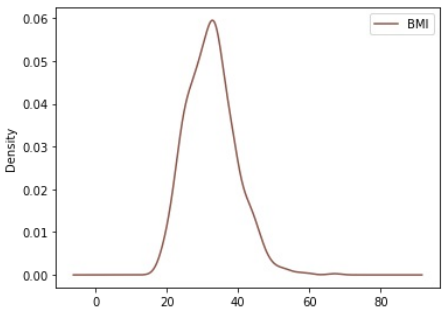


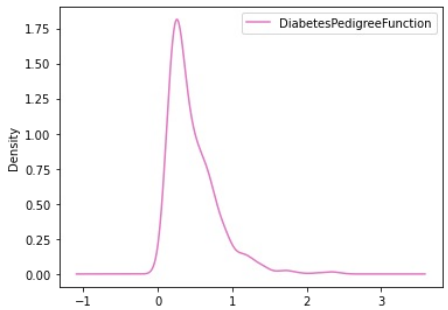
In [13]:
```
df.plot(kind='density', subplots=True, layout=(3,3), figsize=(20, 15), sharex=False)
plt.show()
```

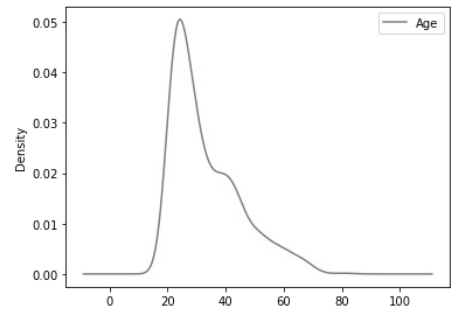


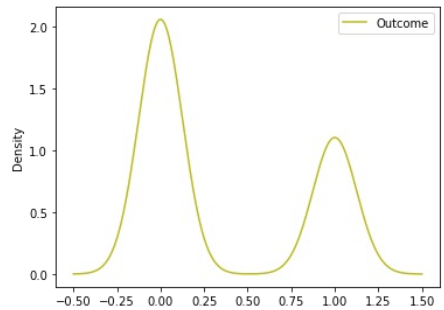








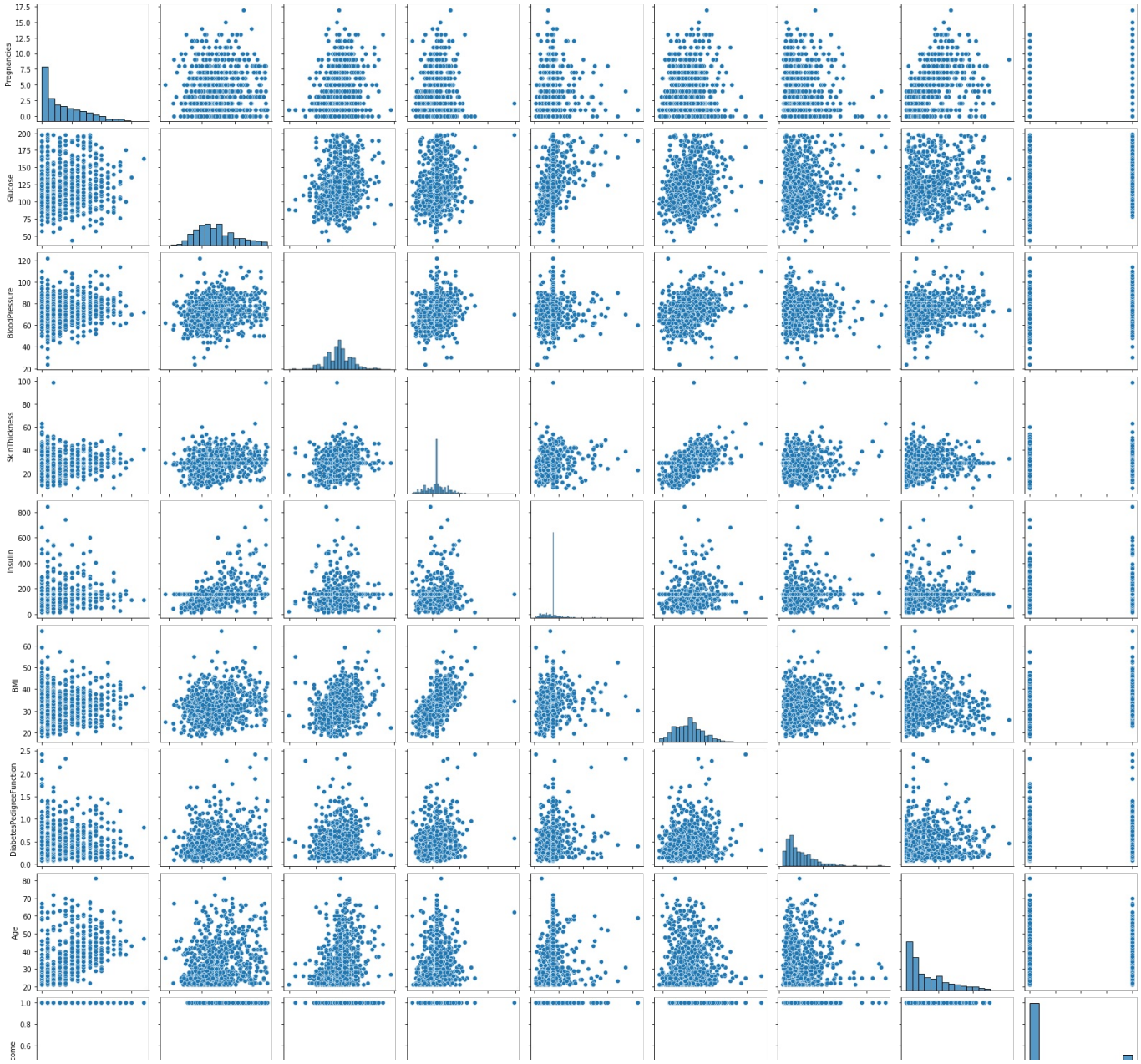


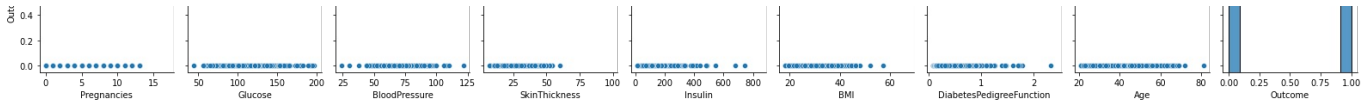


In [14]:

```
# Saçılım(scatter) grafiği
plt.figure(figsize = (20,15))
sns.pairplot(df)
plt.show()
```

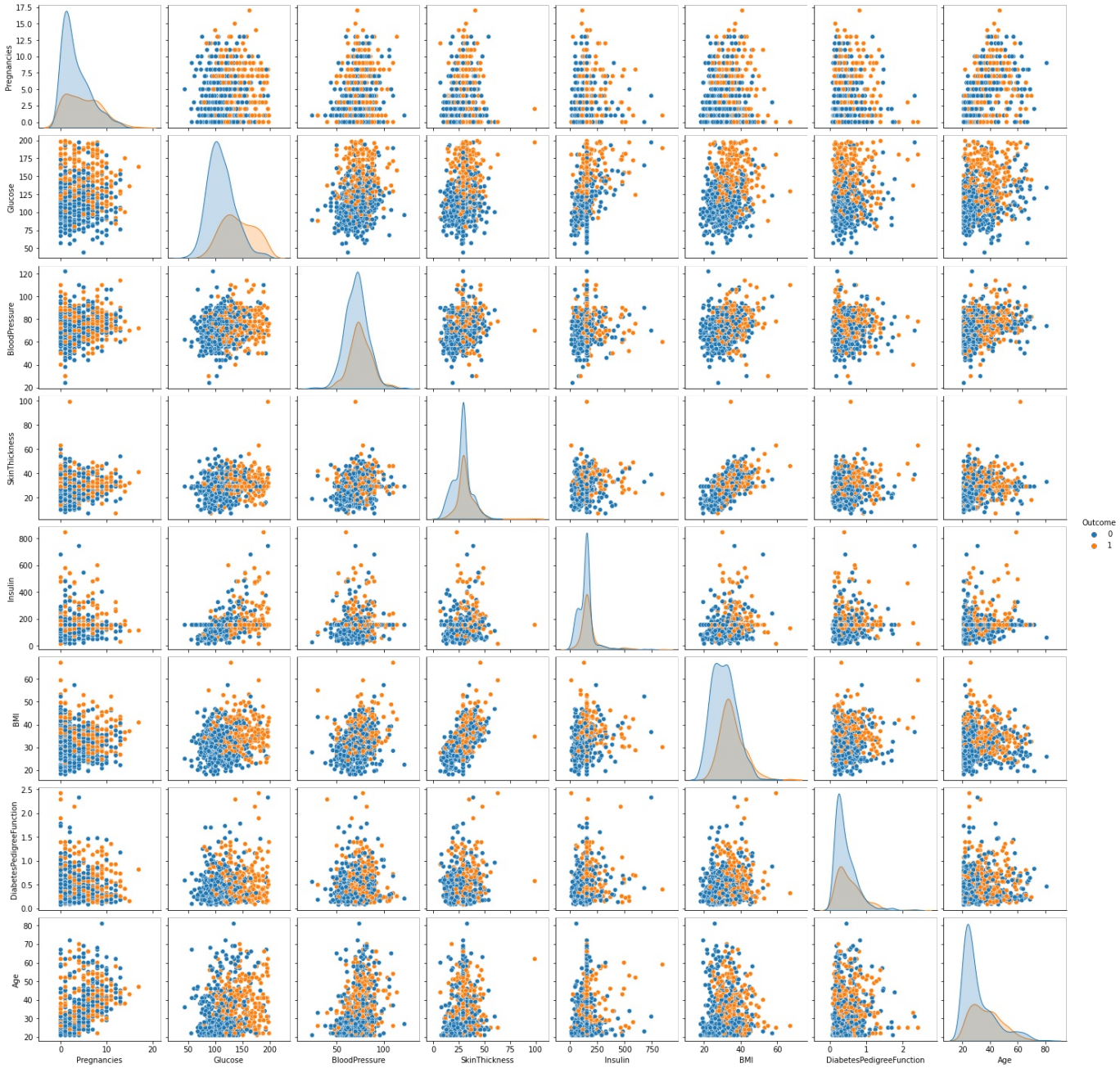
```
<Figure size 1440x1080 with 0 Axes>
```





In [15]:
```
sns.pairplot(df,hue='Outcome')
```

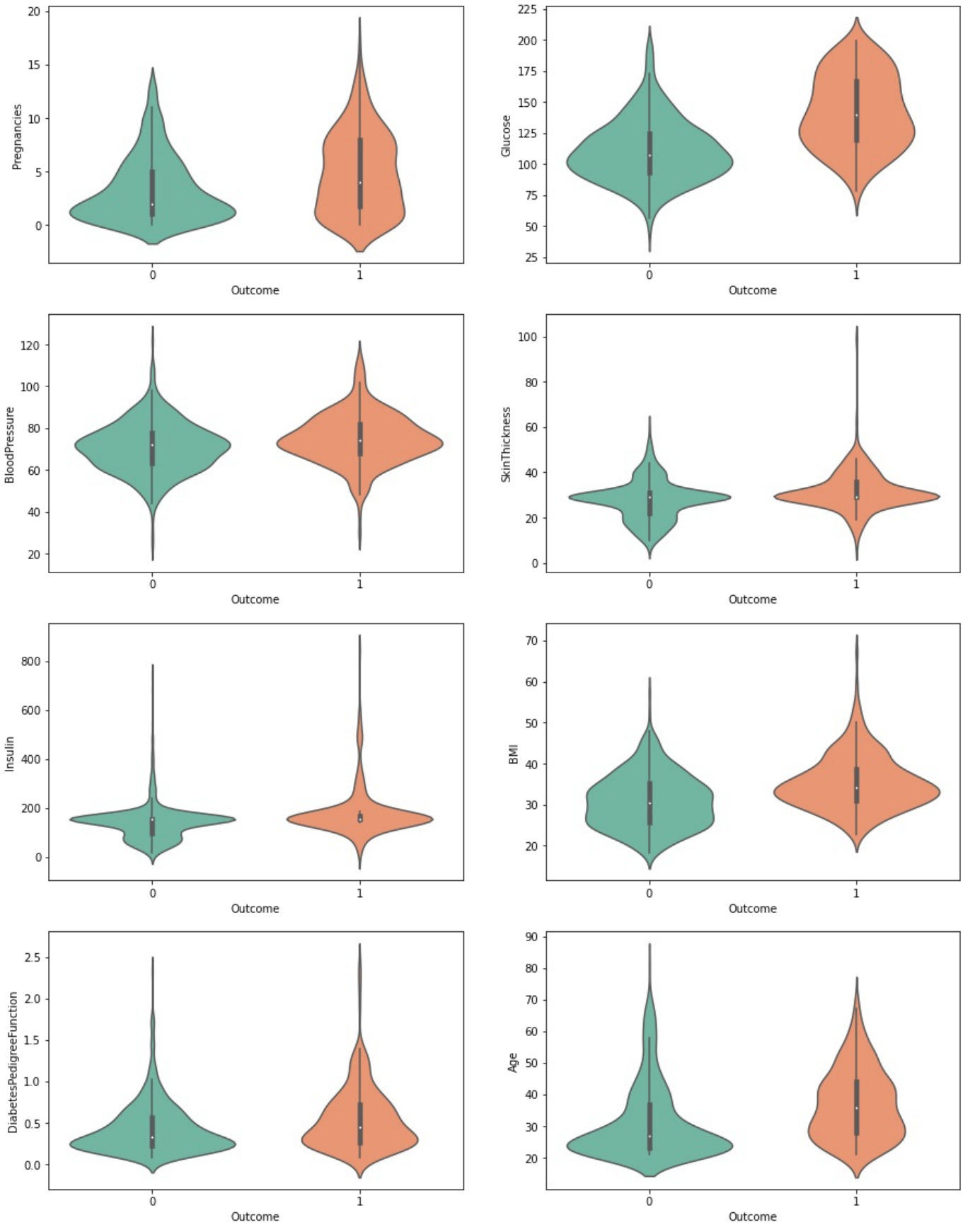
Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x209fe772550>
```



In [16]:
```
# Keman grafiği
plt.figure(figsize=(15,20))

plt.subplot(4,2,1)
sns.violinplot(x='Outcome',y='Pregnancies', data=df,palette='Set2')

plt.subplot(4,2,2)
sns.violinplot(x='Outcome',y='Glucose', data=df,palette='Set2')

plt.subplot(4,2,3)
sns.violinplot(x='Outcome',y='BloodPressure', data=df,palette='Set2')

plt.subplot(4,2,4)
sns.violinplot(x='Outcome',y='SkinThickness', data=df,palette='Set2')

plt.subplot(4,2,5)
sns.violinplot(x='Outcome',y='Insulin', data=df,palette='Set2')
```

```
plt.subplot(4,2,6)
sns.violinplot(x='Outcome',y='BMI', data=df,palette='Set2')

plt.subplot(4,2,7)
sns.violinplot(x='Outcome',y='DiabetesPedigreeFunction', data=df,palette='Set2')

plt.subplot(4,2,8)
sns.violinplot(x='Outcome',y='Age', data=df,palette='Set2')
```

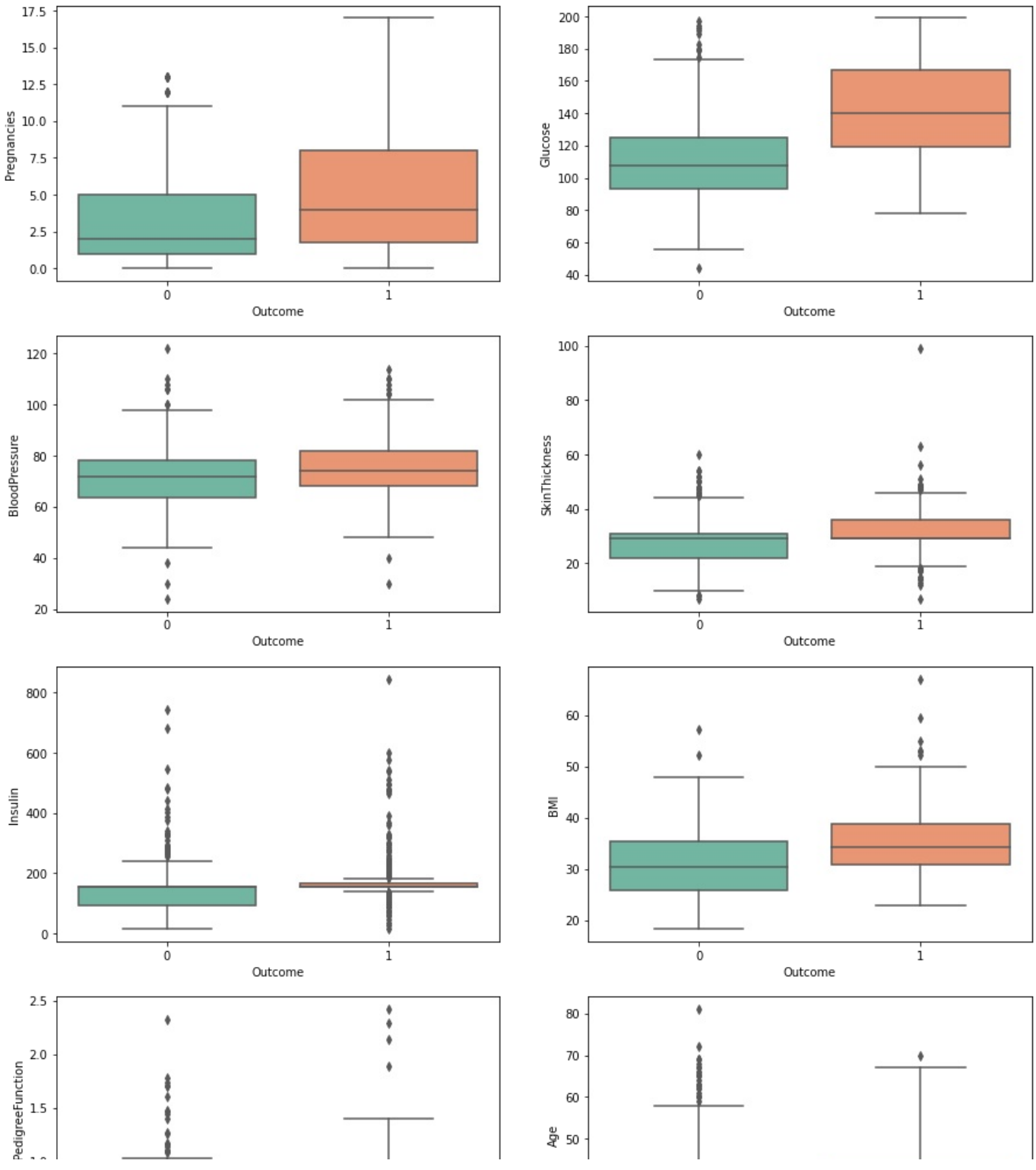
Out[16]: <AxesSubplot:xlabel='Outcome', ylabel='Age'>

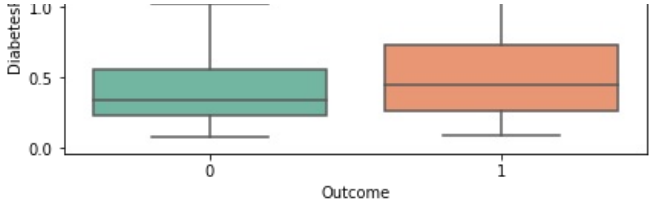


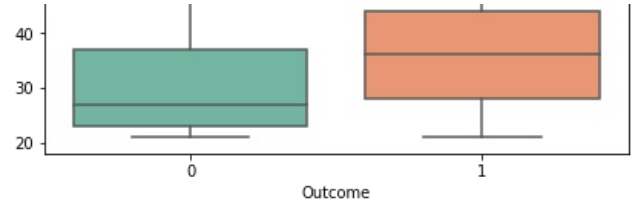
In [17]:
```
# Kutu grafiği
plt.figure(figsize=(15,20))
```

```
plt.subplot(4,2,1)
sns.boxplot(x='Outcome',y='Pregnancies', data=df,palette='Set2')

plt.subplot(4,2,2)
sns.boxplot(x='Outcome',y='Glucose', data=df,palette='Set2')

plt.subplot(4,2,3)
sns.boxplot(x='Outcome',y='BloodPressure', data=df,palette='Set2')

plt.subplot(4,2,4)
sns.boxplot(x='Outcome',y='SkinThickness', data=df,palette='Set2')

plt.subplot(4,2,5)
sns.boxplot(x='Outcome',y='Insulin', data=df,palette='Set2')

plt.subplot(4,2,6)
sns.boxplot(x='Outcome',y='BMI', data=df,palette='Set2')

plt.subplot(4,2,7)
sns.boxplot(x='Outcome',y='DiabetesPedigreeFunction', data=df,palette='Set2')

plt.subplot(4,2,8)
sns.boxplot(x='Outcome',y='Age', data=df,palette='Set2')
```

Out[17]: <AxesSubplot:xlabel='Outcome', ylabel='Age'>







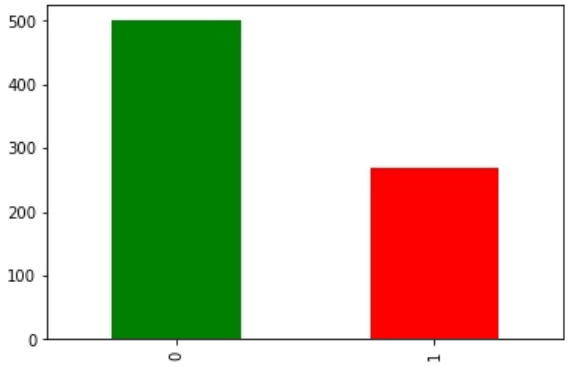
In [18]:

```
c = ['green', 'red']
print(df.Outcome.value_counts())
df.Outcome.value_counts().plot(kind="bar", color = c)
```

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

Out[18]:

```
<AxesSubplot:>
```



Korelasyon Analizi

In [19]:

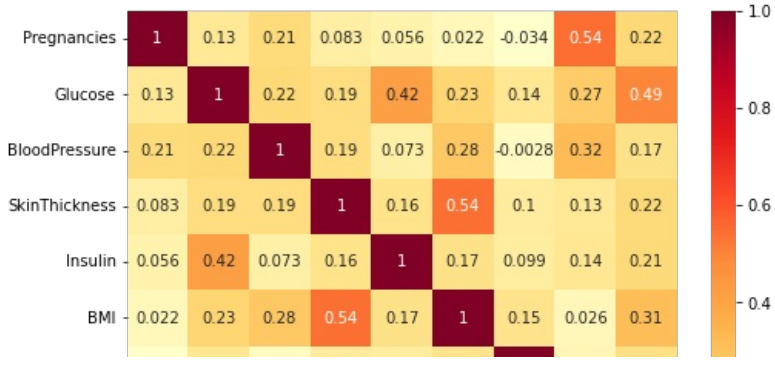
```
df.corr()
```

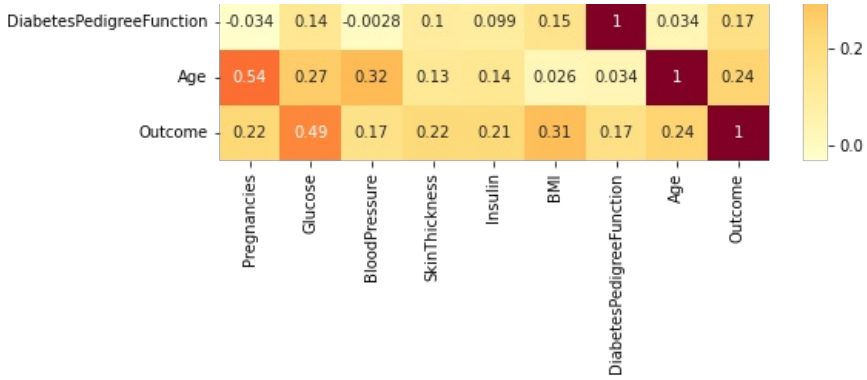
Out[19]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outco
Pregnancies	1.000000	0.127911	0.208522	0.082989	0.056027	0.021565	-0.033523	0.544341	0.221
Glucose	0.127911	1.000000	0.218367	0.192991	0.420157	0.230941	0.137060	0.266534	0.492
BloodPressure	0.208522	0.218367	1.000000	0.192816	0.072517	0.281268	-0.002763	0.324595	0.166
SkinThickness	0.082989	0.192991	0.192816	1.000000	0.158139	0.542398	0.100966	0.127872	0.215
Insulin	0.056027	0.420157	0.072517	0.158139	1.000000	0.166586	0.098634	0.136734	0.214
BMI	0.021565	0.230941	0.281268	0.542398	0.166586	1.000000	0.153400	0.025519	0.311
DiabetesPedigreeFunction	-0.033523	0.137060	-0.002763	0.100966	0.098634	0.153400	1.000000	0.033561	0.173
Age	0.544341	0.266534	0.324595	0.127872	0.136734	0.025519	0.033561	1.000000	0.238
Outcome	0.221898	0.492928	0.166074	0.215299	0.214411	0.311924	0.173844	0.238356	1.000

In [20]:

```
# Heatmap
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(),annot=True,cmap='YlOrRd')
plt.show()
```





In [21]:
```
# Korelasyonu en yüksek 4 değerini alıyoruz
df.corr().nlargest(4,'Outcome').index
```

Out[21]:
```
Index(['Outcome', 'Glucose', 'BMI', 'Age'], dtype='object')
```

Lojistik regresyon

In [22]:
```
from sklearn import linear_model
from sklearn.model_selection import cross_val_score
```

In [23]:
```
x = df[['Glucose', 'BMI', 'Age']]
y = df.iloc[:,8]
y
```

Out[23]:
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [24]:
```
log_reg = linear_model.LogisticRegression()
```

In [25]:
```
log_reg_score = cross_val_score(log_reg,x,y,cv = 10,scoring='accuracy').mean()
```

In [26]:
```
log_reg_score
```

Out[26]:
```
0.7669856459330144
```

SVM

In [27]:
```
from sklearn import svm
```

In [28]:
```
linear_svm = svm.SVC(kernel = 'linear')
```

In [29]:
```
linear_svm_score = cross_val_score(linear_svm,x,y,cv = 10,scoring='accuracy').mean()
```

In [30]:
```
linear_svm_score
```

Out[30]:
```
0.7656527682843473
```

bu iki algoritma içerisinde aralarında küçük fark olmasına rağmen daha yüksek değere sahip olan lojistik regresyonu tercih ediyoruz.

In [31]:
```
# Modeli kaydetmek için pickle kütüphanesini import ediyoruz
import pickle
```

In [32]:
```
filename = 'diabetes.sav'
```

In [33]:
```
log_reg.fit(x,y)
pickle.dump(log_reg,open(filename,'wb'))
```

In [34]:
```
# Modelin çağırılması
loaded_model = pickle.load(open(filename,'rb'))
```

In [35]:
```
loaded_model
```

Out[35]:
```
LogisticRegression()
```

In [36]:
```
# Tahmin yapalım
Glucose = 70
BMI = 60
Age = 50
prediction = loaded_model.predict([[Glucose,BMI,Age]])
```

In [37]:
```
prediction
```

Out[37]:
```
array([1], dtype=int64)
```

In [38]:
```
Glucose = 80
BMI = 30
Age = 65
prediction = loaded_model.predict([[Glucose,BMI,Age]])
```

In [39]:
```
prediction
```

Out[39]:
```
array([0], dtype=int64)
```

In [ ]:

In [ ]: