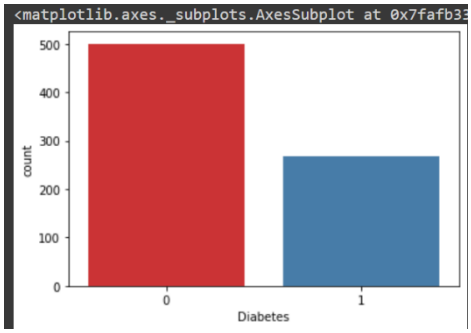
Öncelikle ‘Diabetes’ sütununu inceledim.

Ham verimizde 268 adet Pozitif, 500 adet Negatif değer bulunuyor.




```
# diabetes sütunu incelemesi
class_counts = df["Diabetes"].value_counts()
class_ratio = class_counts[1] / class_counts[0]
print(class_ratio)
```

```
0.536
```

```
class_counts[1]
```

```
268
```

```
class_counts[0]
```

```
500
```

Daha sonra veri ön işleme adımına, eksik değerlerle ilgilenmeye geçtim.

"Pregnancies", "Diabetes" sütunlarını hariç tutarak diğer sütunlardaki 0 değerlerini NaN tipine çevirdim.

Bu işlemden sonra eksik verilerin kaç adet olduğunu hesapladığımda şu sonucu aldım:

```
Pregnancies                 0
Glucose                     5
BP                         35
SkinThickness             227
Insulin                   374
BMI                        11
PedigreeFunc                0
Age                         0
Diabetes                    0
dtype: int64
```

```
Pregnancies              0.000000
Glucose                  0.651042
BP                       4.557292
SkinThickness           29.557292
Insulin                 48.697917
BMI                      1.432292
PedigreeFunc             0.000000
Age                      0.000000
Diabetes                 0.000000
dtype: float64
```

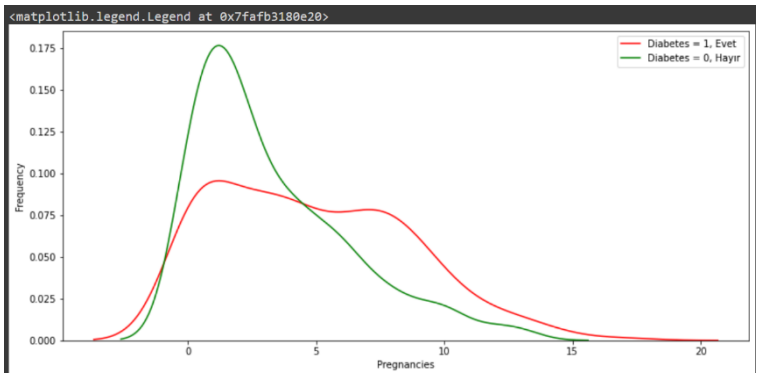
Tuple silme veya hangi yöntemle doldurma yapacağım gibi konularda fikir vermesi açısından, eksik verileri ondalık şekilde göstererek her özniteliğin %kaç verisi eksik olduğunu hesapladım

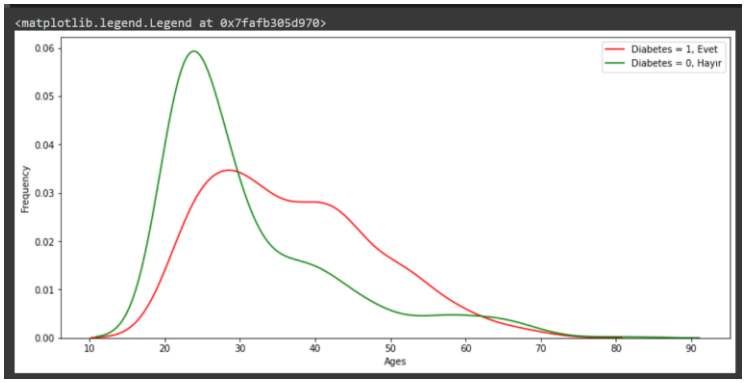
Glucose, BP, BMI, SkinThickness ve Insulin sütunlarındaki eksik değerleri doldurdum.Glucose ve BP için NaN değerleri ortalama yöntemiyle; BMI, SkinThickness ve Insulin için median yöntemiyle değiştirdim.

```
df['Glucose'] = df['Glucose'].replace(np.nan, df['Glucose'].mean())

df['BP'] = df['BP'].replace(np.nan, df['BP'].mean())

df['BMI'] = df['BMI'].replace(np.nan, df['BMI'].median())

df['SkinThickness'] = df['SkinThickness'].replace(np.nan, df['SkinThickness'].median())

df['Insulin'] = df['Insulin'].replace(np.nan, df['Insulin'].median())
```

```
Pregnancies              0
Glucose                  0
BP                       0
SkinThickness            0
Insulin                  0
BMI                      0
PedigreeFunc             0
Age                      0
Diabetes                 0
dtype: int64
```

Solda bağımsız değişken Pregnancies’in, sağda Age’in Diabets sonuç değişkenine etkisini plot çizdirerek görselleştirdim.





df.describe().T

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.000000	6.00000	17.00
Glucose	768.0	121.686763	30.435949	44.000	99.75000	117.000000	140.25000	199.00
BP	768.0	72.405184	12.096346	24.000	64.00000	72.202592	80.00000	122.00
SkinThickness	768.0	29.108073	8.791221	7.000	25.00000	29.000000	32.00000	99.00
Insulin	768.0	140.671875	86.383060	14.000	121.50000	125.000000	127.25000	846.00
BMI	768.0	32.455208	6.875177	18.200	27.50000	32.300000	36.60000	67.10
PedigreeFunc	768.0	0.471876	0.331329	0.078	0.24375	0.372500	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.000000	41.00000	81.00
Diabetes	768.0	0.348958	0.476951	0.000	0.00000	0.000000	1.00000	1.00

Veri setini incelediğimde bir özniteliğin ortalaması 0.3 iken bir diğerinin 140 olduğunu görüp normalizasyon yapmayı tercih ettim. Bunun için Min Max yöntemini kullandım.

```
# Normalizasyon : Min-Max
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
df_normalizasyon = sc.fit_transform(df)
df_normalizasyon = pd.DataFrame(df_normalizasyon)
```

Ve ön işleme adımını burada sonlandırıp sınıflandırmaya geçtim.

Sınıflandırma için yaptığım ek araştırmalar ışığında seçenekler arasından veri setime en uygun olduğunu öngördüğüm Bayes ve Random Forests sınıflandırma algoritmalarını tercih ettim.

Bağımlı değişkenler; Pregnancies ve Age' i dikkate alarak veri setimi böldüm. Test seti büyüklüğünü %30 belirledim.

```
# öznitelik seçimi - [Pregnancies, Age]
X = df_normalizasyon.iloc[:, [0, 7]].values
Y = df_normalizasyon.iloc[:, 8].values
```

```
X_train boyutu  (537, 2)
X_test boyutu  (231, 2)
Y_train boyutu  (537,)
Y_test boyutu  (231,)
```

```
# test train x y ayırma test kümesi boyutu %30
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.30, random_state = 42, stratify = df['Diabetes'] )
```

Tüm bu işlemler sonucunda sınıflandırma performansları aşağıdaki gibi oldu.

```
Random Forest Sınıflama Performansı:
Accuracy -  61.904761904761905
Precision -  44.776119402985074
Recall -  37.03703703703704
```

```
Bayes Sınıflama Performansı:
Accuracy -  67.09956709956711
Precision -  55.10204081632652
Recall -  33.33333333333333
```