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

In [2]:
```python
dataset = pd.read_csv('health care diabetes.csv')
```

In [3]:
```python
dataset.head()
```

Out[3]:

	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 [4]:
```python
dataset.shape
```

Out[4]: (768, 9)

In [5]:
```python
dataset.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 [6]:
```python
dataset.isna().sum().sum()
```

Out[6]: 0

In [7]:
```python
dataset.nunique()
```

Out[7]:
```
Pregnancies                  17
Glucose                     136
BloodPressure                47
SkinThickness                51
Insulin                     186
BMI                         248
DiabetesPedigreeFunction    517
Age                          52
Outcome                       2
dtype: int64
```

Observations

All variables are integer or float types. There are no missing values in the dataset. Outcome is our Target Variable.

In [8]:
```python
dataset.describe().T
```

Out[8]:

	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

Observations

*Data set contains women with an average of 4 pregnancies and maximum of 17.*Features like Glucose, BloodPressure, SkinThickness, and Insulin have minimum values as 0 which might be data input errors and should explore it further. *Difference between maximum value for features like SkinThickness, Insulin and Age and 3rd quartile which suggest ,there might be outliers present in the data.*Average age of women in the dataset is 33 years and median is 29.

EXPLORATORY DATA ANALYSIS

In [9]:
```python
def dist_box(data):
    Name=data.name.upper()
    fig,(ax_box,ax_dis)  =plt.subplots(nrows=2,sharex=True,gridspec_kw = {"height_ratios": (
.25, .75)},figsize=(8, 5))
    mean=data.mean()
    median=data.median()
    mode=data.mode().tolist()[0]

    sns.set_palette(sns.color_palette("Set1", 8))
    fig.suptitle("SPREAD OF DATA FOR "+ Name  , fontsize=18, fontweight='bold')
    sns.boxplot(x=data,showmeans=True, orient='h',ax=ax_box)
    ax_box.set(xlabel='')
    sns.despine(top=True,right=True,left=True)
    sns.set_palette(sns.color_palette("Set1", 8))
    sns.distplot(data,kde=False,ax=ax_dis)
    ax_dis.axvline(mean, color='r', linestyle='--',linewidth=2)
    ax_dis.axvline(median, color='g', linestyle='-',linewidth=2)
    ax_dis.axvline(mode, color='y', linestyle='-',linewidth=2)
    plt.legend({'Mean':mean,'Median':median,'Mode':mode})
```

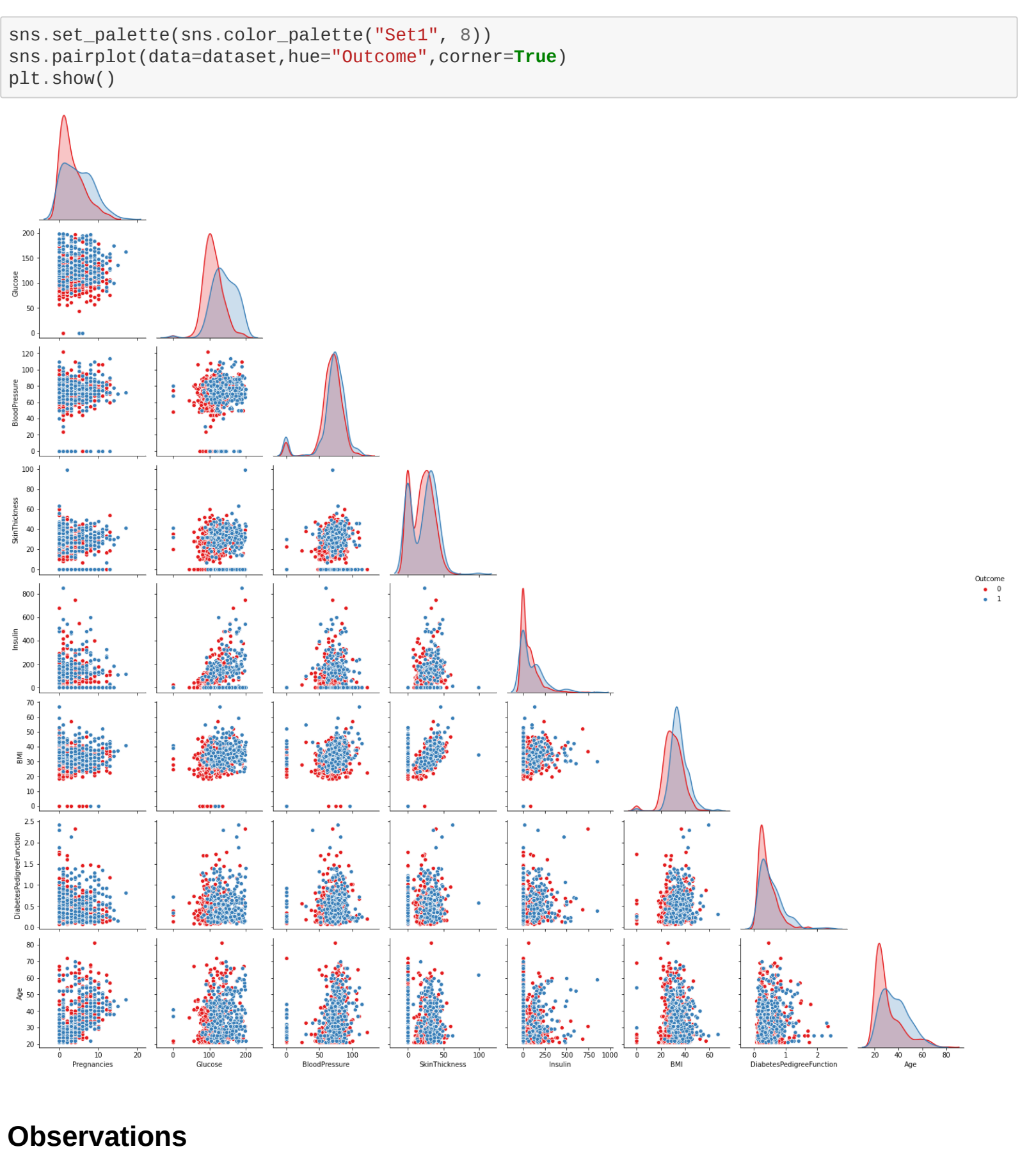
In [10]:
```python
#select all quantitative columns for checking the spread
list_col= dataset.select_dtypes(include='number').columns.to_list()
for i in range(len(list_col)):
    dist_box(dataset[list_col[i]])
```

Observations:

*Number of pregnancies is right-skewed.The boxplot shows that there are few outliers to the right.*Plasma glucode is normally distrubted.0 value is an outlier for this variable. *The distribution for blood pressure looks fairy normal except few outliers evident from the boxplot. We can see are some observations with 0 blood pressure but a 0 value of blood pressure is not possible and we should treat the 0 value as missing data.Most of the women have normal blood pressure.*There are one extreme value of 99 in Skin thickness, many value with 0 value of skin thicknessand we should treat the 0 values as missing data. *Insulin is right-skewed. There are some outliers to the right A 0 value in insulin is not possible. We should treat the 0 values as missing data.75% of women have less than 127 mu U/ml of insulin concentration and an average of 80 mu U/ml.*BMI is normally distributed with the mean and median of approximately 32.There are some outliers in this variable. A 0 value in mass is not possible we should treat the 0 values as missing data. *Diabetes pedigree function is skewed to the right and there are some outliers in this variable.*Age is right-skewed.There are outliers on higher end.

In [11]:
```python
sns.set_palette(sns.color_palette("Set1", 8))
plt.figure(figsize=(15,7))
sns.heatmap(dataset.corr(),annot=True,vmin=-1,vmax=1,cmap='viridis')
plt.show()
```

In [12]:
```python
sns.set_palette(sns.color_palette("Set1", 8))
sns.pairplot(data=dataset,hue="Outcome",corner=True)
plt.show()
```



Observations

*Outcome variable class shows a moderate correlation with 'Glucose'.*There is a positive correlation between age and number of pregnancies which makes sense. *Insulin and skin thickness also shows a moderate positive correlation.*We can see that most non-diabetic persons have glucose concentration<=100 and BMI<30 *However, there are overlapping distributions for diabetic and non-diabetic persons. We should investigate it further.

Handling Missing value

Glucose, Blood pressure, SkinThickness , Insulin ,BMI had 0 values replacing them by median of the respective variable

In [13]:
```python
dataset.loc[dataset.Glucose == 0, 'Glucose'] = dataset.Glucose.median()
dataset.loc[dataset.BloodPressure == 0, 'BloodPressure'] = dataset.BloodPressure.median()
dataset.loc[dataset.SkinThickness == 0, 'SkinThickness'] = dataset.SkinThickness.median()
dataset.loc[dataset.Insulin == 0, 'Insulin'] = dataset.Insulin.median()
dataset.loc[dataset.BMI == 0, 'BMI'] = dataset.BMI.median()
```

In [14]:
```python
X = dataset.drop('Outcome',axis=1)
y = dataset['Outcome']
```

In [15]:
```python
from sklearn.model_selection import train_test_split
```

In [16]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,stra
tify=y)
print(X_train.shape, X_test.shape)
```

(537, 8) (231, 8)

In [17]:
```python
y.value_counts(1)
```

Out[17]:
```
0    0.651042
1    0.348958
Name: Outcome, dtype: float64
```

In [18]:
```python
y_test.value_counts(1)
```

Out[18]:
```
0    0.649351
1    0.350649
Name: Outcome, dtype: float64
```

In [19]:
```python
from sklearn.preprocessing import StandardScaler
```

In [20]:
```python
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Model Building

In [21]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
```

In [22]:
```python
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,plot_confusion_matrix
from sklearn import metrics
from sklearn.metrics import  classification_report, accuracy_score, precision_score, recall_
score,f1_score
```

In [23]:
```python
#Fitting the model
classifier=DecisionTreeClassifier(criterion = 'entropy')
classifier.fit(X_train,y_train)
```

Out[23]: DecisionTreeClassifier(criterion='entropy')

In [24]:
```python
y_pred_dec = classifier.predict(X_test)
```

In [25]:
```python
print(confusion_matrix, accuracy_score)
```

```
<function confusion_matrix at 0x0000021BF9D435E0> <function accuracy_score at 0x0000021BF9D43
4C0>
```

In [26]:
```python
confusion_matrix(y_test,y_pred_dec)
```

Out[26]:
```
array([[119,  31],
       [ 37,  44]], dtype=int64)
```

In [27]:
```python
accuracy_score(y_test,y_pred_dec)
```

Out[27]: 0.7056277056277056

In [28]:
```python
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier(criterion = 'entropy')
classifier.fit(X_train,y_train)
```

Out[28]: RandomForestClassifier(criterion='entropy')

In [29]:
```python
y_pred_ran = classifier.predict(X_test)
```

In [30]:
```python
print(confusion_matrix, accuracy_score)
```

```
<function confusion_matrix at 0x0000021BF9D435E0> <function accuracy_score at 0x0000021BF9D43
4C0>
```

In [31]:
```python
confusion_matrix(y_test,y_pred_ran)
```

Out[31]:
```
array([[128,  22],
       [ 36,  45]], dtype=int64)
```

In [32]:
```python
accuracy_score(y_test,y_pred_ran)
```

Out[32]: 0.7489177489177489

In [33]:
```python
from sklearn.neighbors import KNeighborsClassifier
```

In [34]:
```python
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)
y_pred_Knn=knn.predict(X_test)
```

In [35]:
```python
print(confusion_matrix, accuracy_score)
```

```
<function confusion_matrix at 0x0000021BF9D435E0> <function accuracy_score at 0x0000021BF9D43
4C0>
```

In [36]:
```python
confusion_matrix(y_test,y_pred_Knn)
```

Out[36]:
```
array([[125,  25],
       [ 37,  44]], dtype=int64)
```

In [37]:
```python
accuracy_score(y_test,y_pred_Knn)
```

Out[37]: 0.7316017316017316

Observations

- accuracy score with KNN is better than Decission Tree classifier but same result with Random Forest classifier

In [38]:
```python
from sklearn import metrics
```

In [39]:
```python
y_pred_proba = classifier.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```


In [ ]: