In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
import plotly.express as px
import plotly.io as pio
```

In [2]:
```
df=pd.read_csv("diabetes.csv")
```

In [3]:
```
df
```

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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [4]:
```
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 [5]:
```
df.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [6]:
```
for i in ["Glucose","BloodPressure","SkinThickness","Insulin","BMI"]:
    df[i].replace(0,np.nan,inplace=True)
    df[i].fillna(df[i].mean(),inplace=True)
```

In [7]:
```
df.describe()
```

Out[7]:

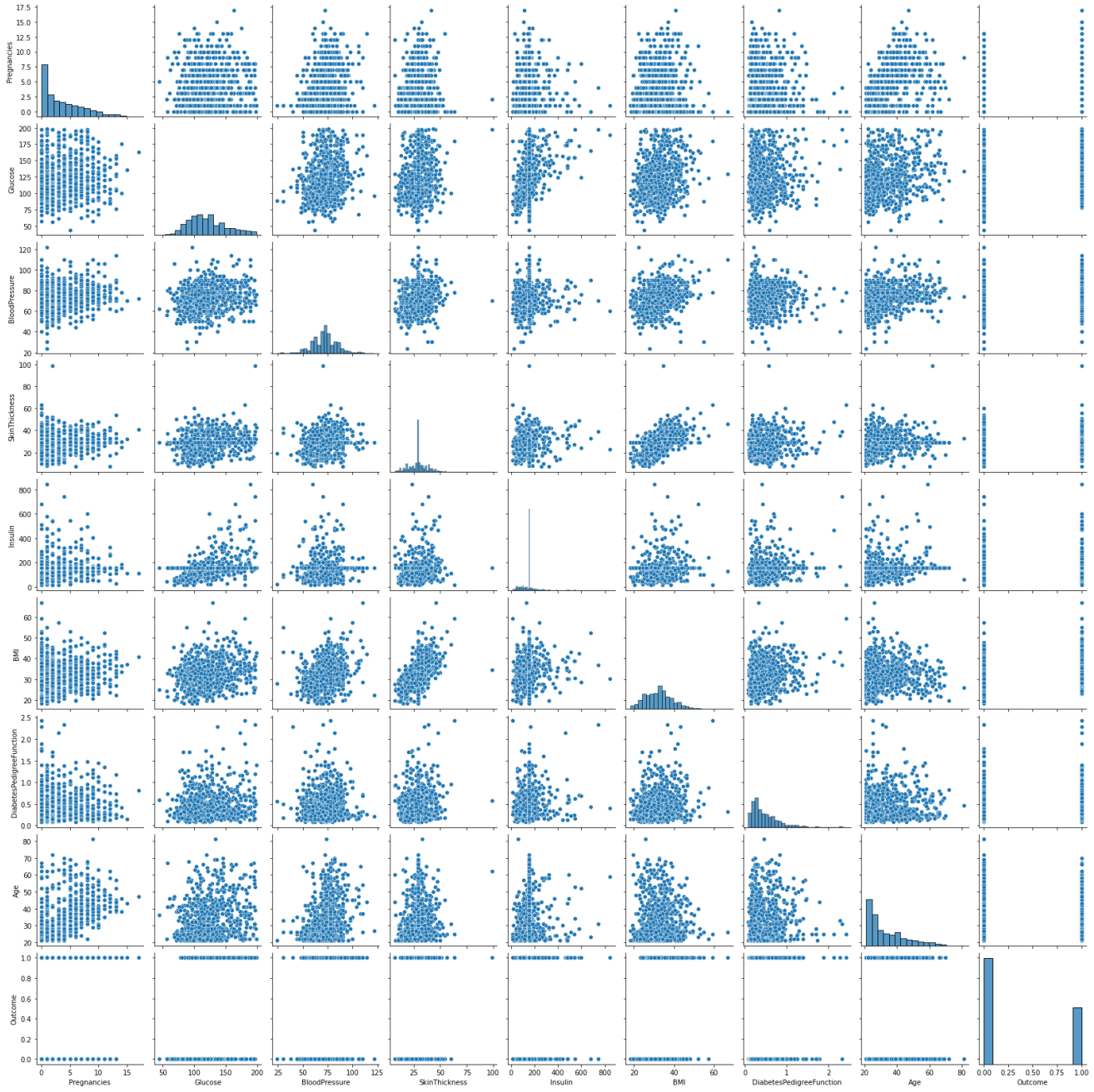
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	121.686763	72.405184	29.153420	155.548223	32.457464	0.471876	33.240885	0.348958
std	3.369578	30.435949	12.096346	8.790942	85.021108	6.875151	0.331329	11.760232	0.476951
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.750000	64.000000	25.000000	121.500000	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.202592	29.153420	155.548223	32.400000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	155.548223	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [8]:
```
df["Glucose"].value_counts()
```

Out[8]:
```
99.0     17
100.0    17
111.0    14
129.0    14
125.0    14
         ..
191.0     1
177.0     1
44.0      1
62.0      1
190.0     1
Name: Glucose, Length: 136, dtype: int64
```

In [9]:
```
sns.pairplot(df)
```

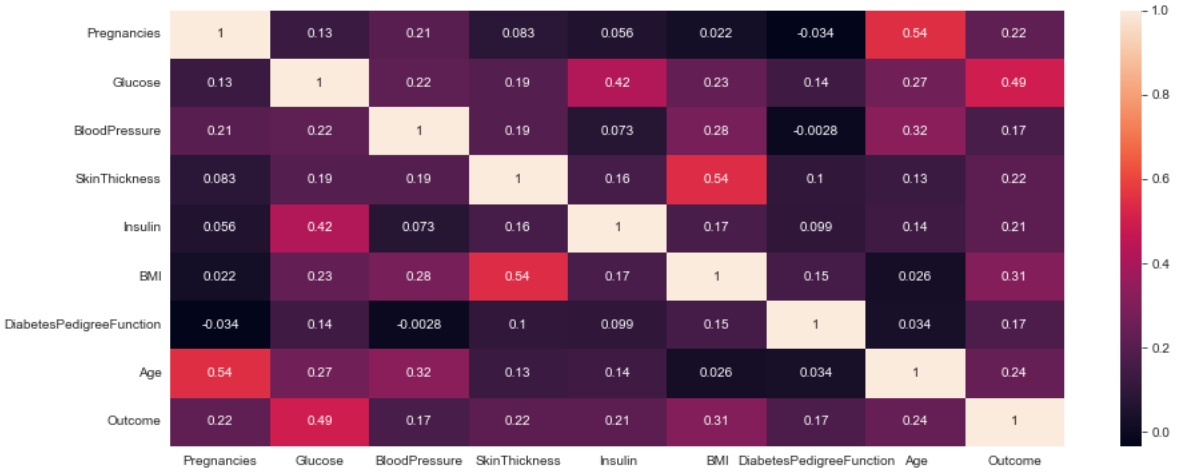
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1856983f880>
```



In [10]:
```
df.isnull().sum()
```

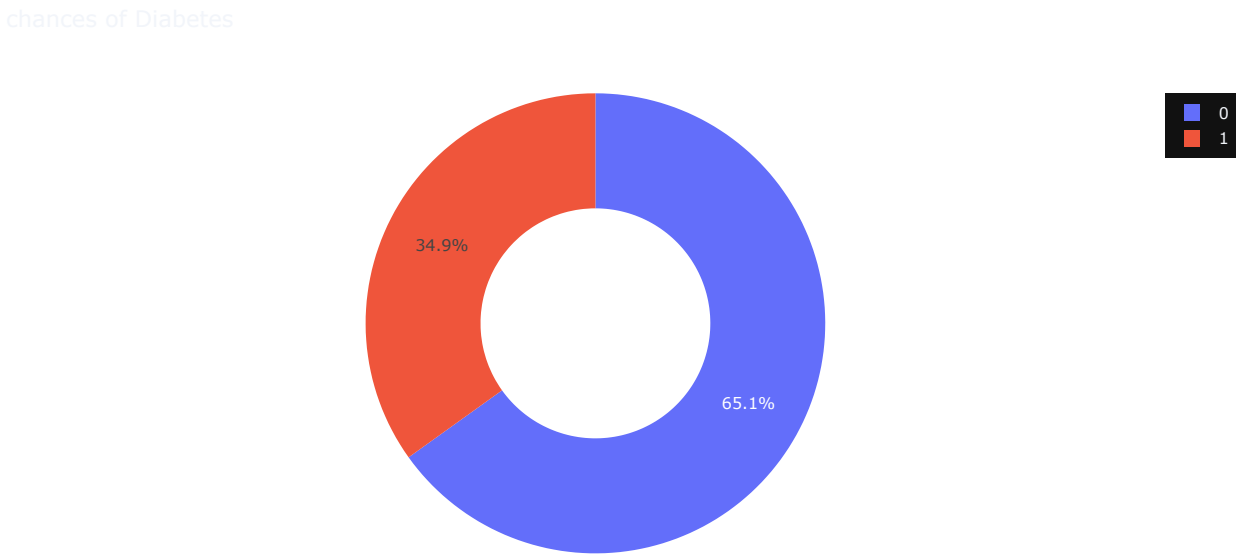
Out[10]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [11]:
```
plt.figure(figsize= (15,6))
sns.set_style("darkgrid")
sns.heatmap(df.corr(),annot= True)
plt.show()
```

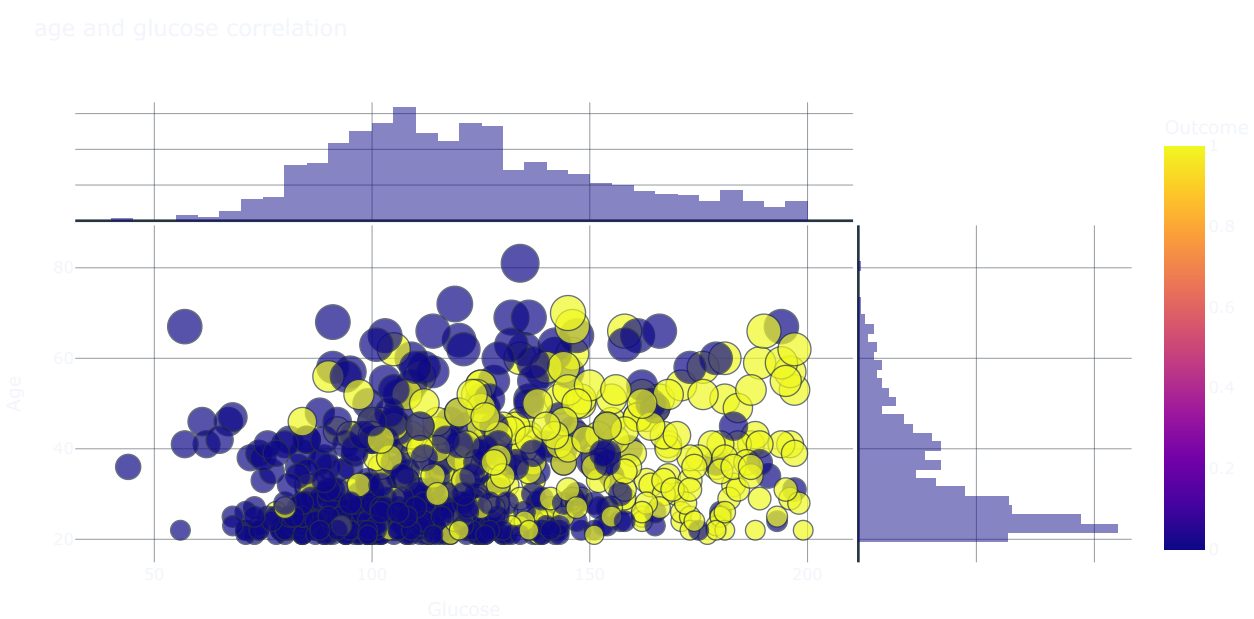


In [12]:
```
ax= px.histogram(df,x= "Age", template= "plotly_dark",color= "Outcome",title='Age distribution')
ax.show()
```

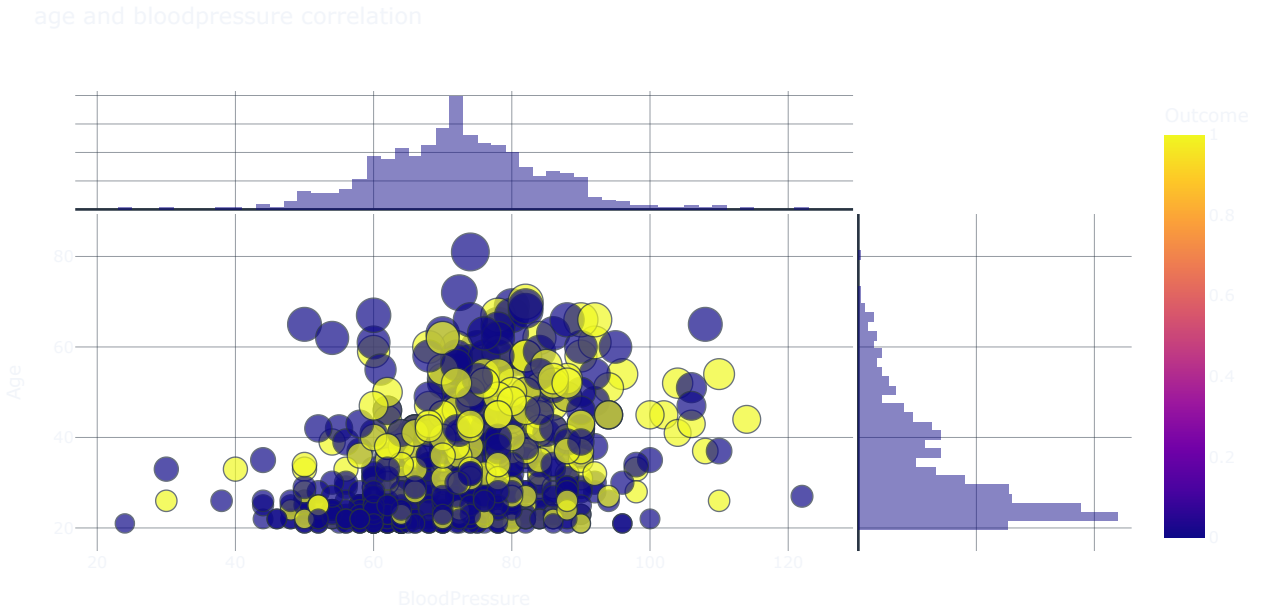
In [13]:
```
ax= px.pie(df, names= "Outcome",template= "plotly_dark",title= "chances of Diabetes",hole= 0.5)
ax.show()
```



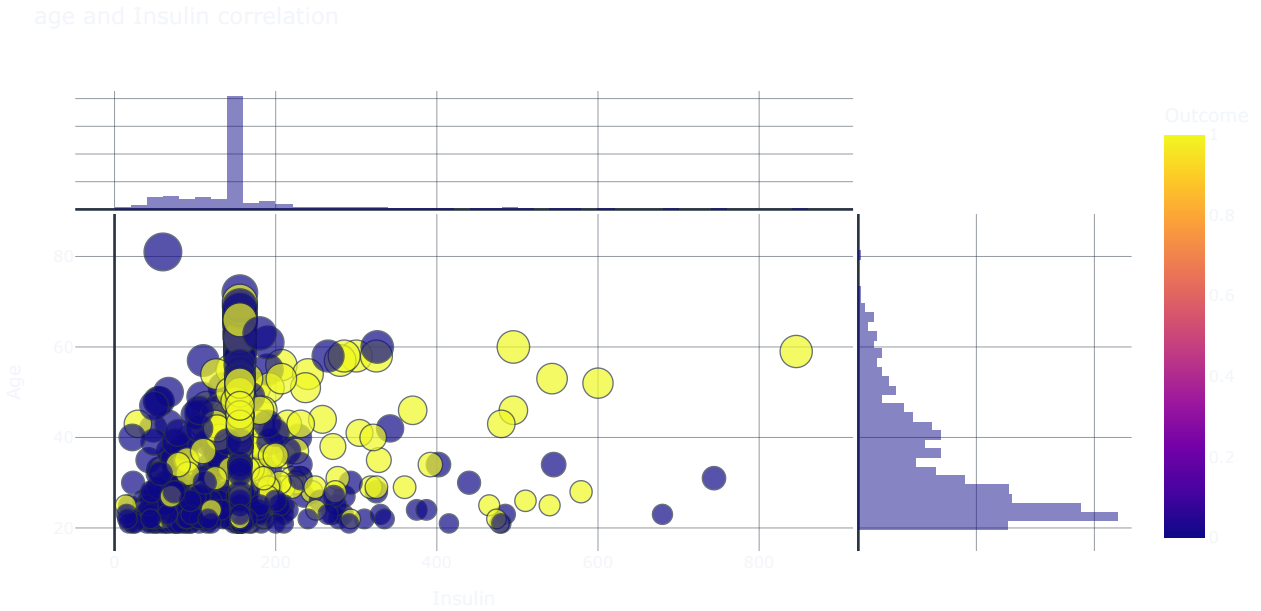
In [14]:
```
ax= px.scatter(df,x= "Glucose",y= "Age",marginal_x='histogram', marginal_y='histogram',size="Age", size_max=20,
               template= "plotly_dark",color= "Outcome",title="age and glucose correlation")
ax.show()
```



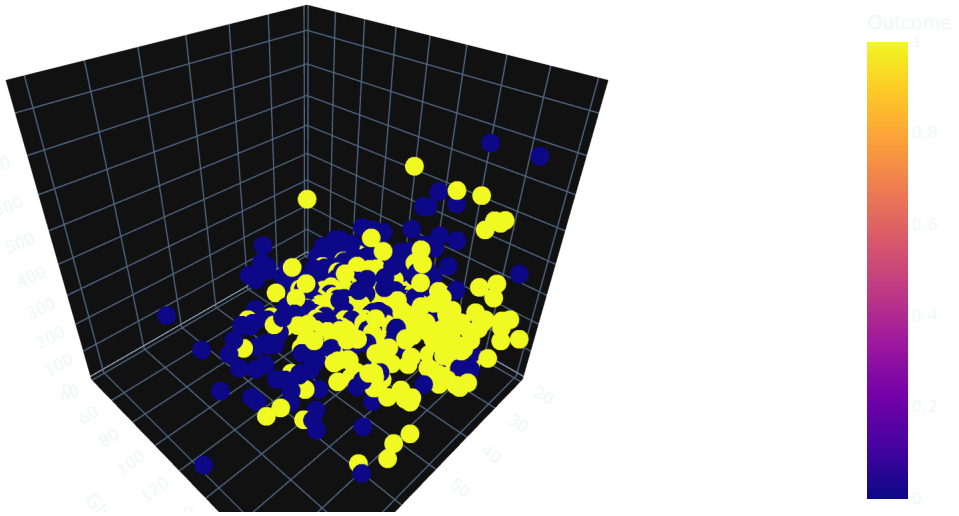
In [15]:
```
ax= px.scatter(df,x= "BloodPressure",y= "Age",marginal_x='histogram', marginal_y='histogram',size="Age", size_max=20,
               template= "plotly_dark",color= "Outcome",title="age and bloodpressure correlation")
ax.show()
```



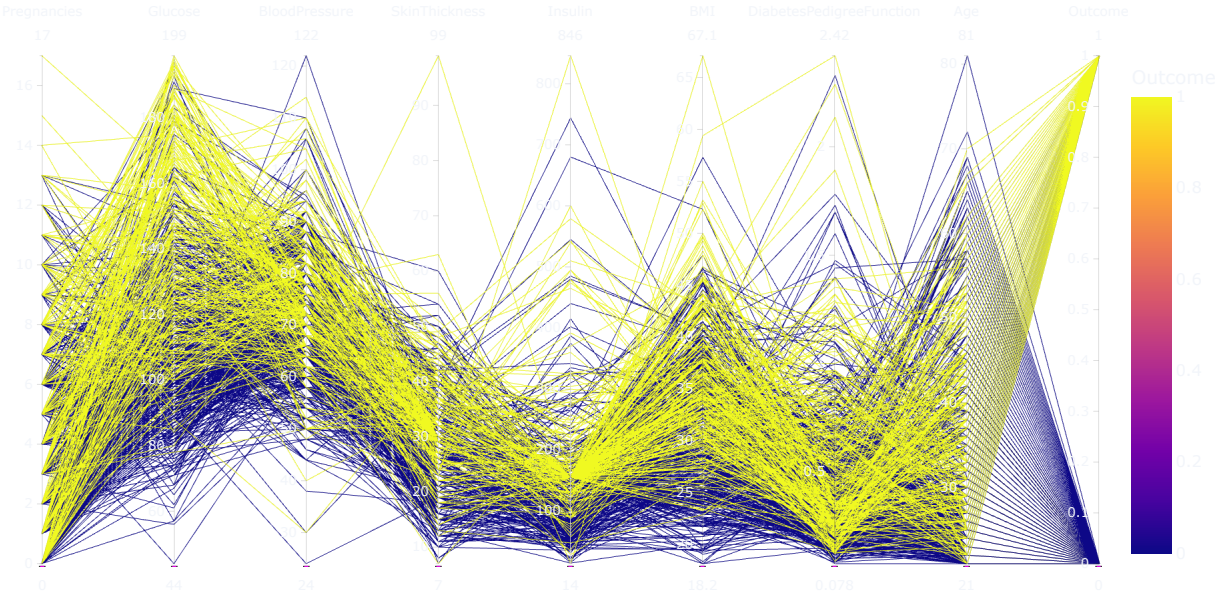
In [16]:
```
ax= px.scatter(df,x= "Insulin",y= "Age",marginal_x='histogram', marginal_y='histogram',size="Age", size_max=20,
               template= "plotly_dark",color= "Outcome",title="age and Insulin correlation")
ax.show()
```



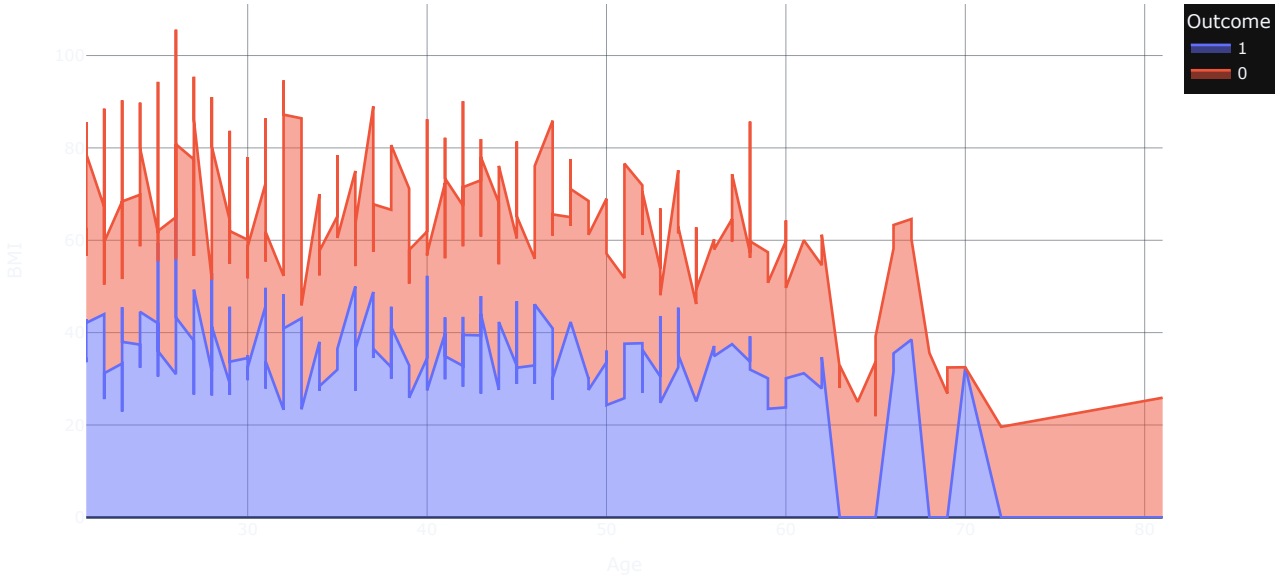
In [17]:
```
ax = px.scatter_3d(df, x="Age", y="Glucose", z="Insulin",template= "plotly_dark",color="Outcome")
ax.show()
```



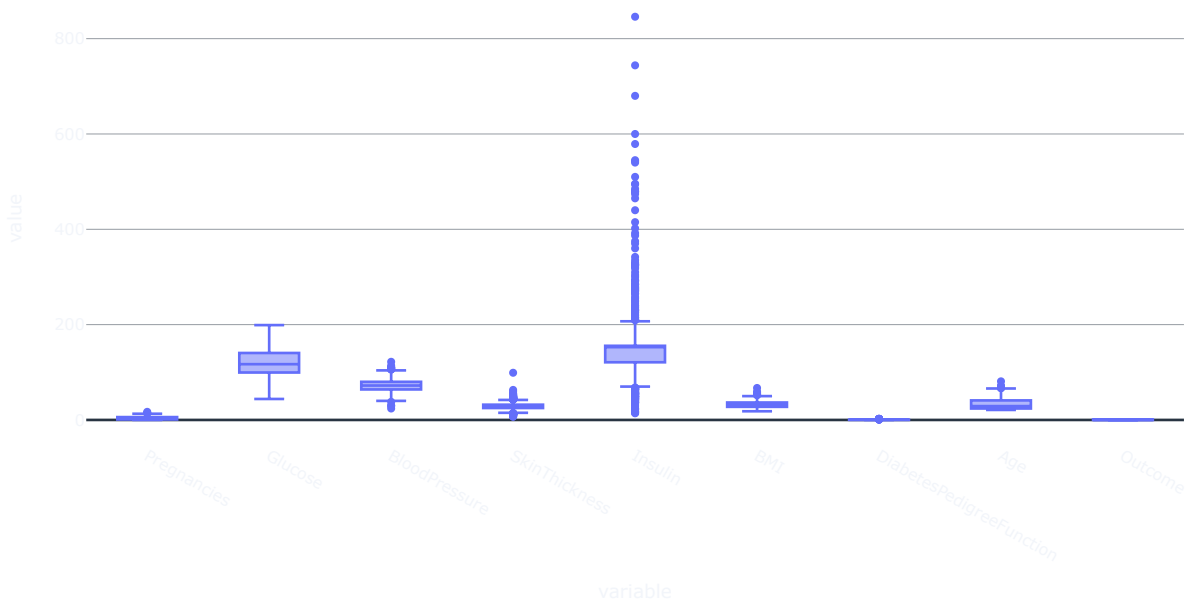
In [19]:
```
ax= px.parallel_coordinates(df, color="Outcome",template= "plotly_dark")
ax.show()
```



In [20]:
```
fig = px.area(df, x="Age", y="BMI", color="Outcome",template= "plotly_dark")
fig.show()
```



In [22]:
```
ax = px.box(df,template= "plotly_dark")
ax.show()
```



In [23]:
```
#splittig x and y for predictoions
```

In [26]:
```
x=df.iloc[:,1:-1]
```

In [27]:
```
x
```

Out[27]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	148.0	72.0	35.00000	155.548223	33.6	0.627	50
1	85.0	66.0	29.00000	155.548223	26.6	0.351	31
2	183.0	64.0	29.15342	155.548223	23.3	0.672	32
3	89.0	66.0	23.00000	94.000000	28.1	0.167	21
4	137.0	40.0	35.00000	168.000000	43.1	2.288	33
...	...	...	...	...	...	...	...
763	101.0	76.0	48.00000	180.000000	32.9	0.171	63
764	122.0	70.0	27.00000	155.548223	36.8	0.340	27
765	121.0	72.0	23.00000	112.000000	26.2	0.245	30
766	126.0	60.0	29.15342	155.548223	30.1	0.349	47
767	93.0	70.0	31.00000	155.548223	30.4	0.315	23

768 rows × 7 columns

In [28]:
```
y=df["Outcome"]
```

In [29]:
```
y
```

Out[29]:
```
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 [30]:
```
#splitting the model in x,y train and test
```

In [31]:
```
from sklearn.model_selection import train_test_split
```

In [33]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.20,random_state=1)
```

In [34]:
```
from sklearn.linear_model import LogisticRegression
```

In [37]:
```
logreg=LogisticRegression()
logreg.fit(xtrain,ytrain)
ypred=logreg.predict(xtest)
```

In [39]:
```
from sklearn.metrics import classification_report
```

In [40]:
```
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.89      0.84        99
           1       0.75      0.60      0.67        55

    accuracy                           0.79       154
   macro avg       0.78      0.74      0.75       154
weighted avg       0.78      0.79      0.78       154
```

In [44]:
```
#feature scaling
```

In [45]:
```
from sklearn.preprocessing import StandardScaler
```

In [51]:
```
sc=StandardScaler()
xtrain=sc.fit_transform(xtrain)
```

In [52]:
```
xtest=sc.transform(xtest)
```

In [53]:
```
#hypertuning using solver parameter
```

In [54]:
```
logreg=LogisticRegression(solver="saga")
logreg.fit(xtrain,ytrain)
ypred=logreg.predict(xtest)
```

In [55]:
```
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.89      0.84        99
           1       0.75      0.60      0.67        55

    accuracy                           0.79       154
   macro avg       0.78      0.74      0.75       154
weighted avg       0.78      0.79      0.78       154
```

In [56]:
```
import pandas_profiling as pp
```

In [57]:
```
pp.ProfileReport(df)
```

Summarize dataset: 100% 82/82 [00:06<00:00, 12.47it/s, Completed]

Generate report structure: 100% 1/1 [00:01<00:00, 1.81s/it]

Render HTML: 100% 1/1 [00:02<00:00, 2.35s/it]

17	1	0.1%
15	1	0.1%
14	2	0.3%
13	10	1.3%
12	9	1.2%
11	11	1.4%
10	24	3.1%
9	28	3.6%
8	38	4.9%
7	45	5.9%

Interactions

Out[57]:

In [ ]: