In [373…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [374…
```
df= pd.read_csv('health care diabetes.csv')
```

In [375…
```
df.head(23)
```

Out[375]:

	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
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1
10	4	110	92	0	0	37.6	0.191	30	0
11	10	168	74	0	0	38.0	0.537	34	1
12	10	139	80	0	0	27.1	1.441	57	0
13	1	189	60	23	846	30.1	0.398	59	1
14	5	166	72	19	175	25.8	0.587	51	1
15	7	100	0	0	0	30.0	0.484	32	1
16	0	118	84	47	230	45.8	0.551	31	1
17	7	107	74	0	0	29.6	0.254	31	1
18	1	103	30	38	83	43.3	0.183	33	0
19	1	115	70	30	96	34.6	0.529	32	1
20	3	126	88	41	235	39.3	0.704	27	0
21	8	99	84	0	0	35.4	0.388	50	0
22	7	196	90	0	0	39.8	0.451	41	1

In [376…
```
df.tail()
```

Out[376]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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

In [377…
```
df.shape
```

Out[377]:
```
(768, 9)
```

In [378…
```
df.dtypes
```

Out[378]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

In [379…
```
df.Outcome.value_counts()
```

Out[379]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [380…
```
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 [381…
```
df.isnull().sum()
```

Out[381]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [382…
```
type(df)
```

Out[382]:
```
pandas.core.frame.DataFrame
```

In [383…
```
df.describe()
```

Out[383]:

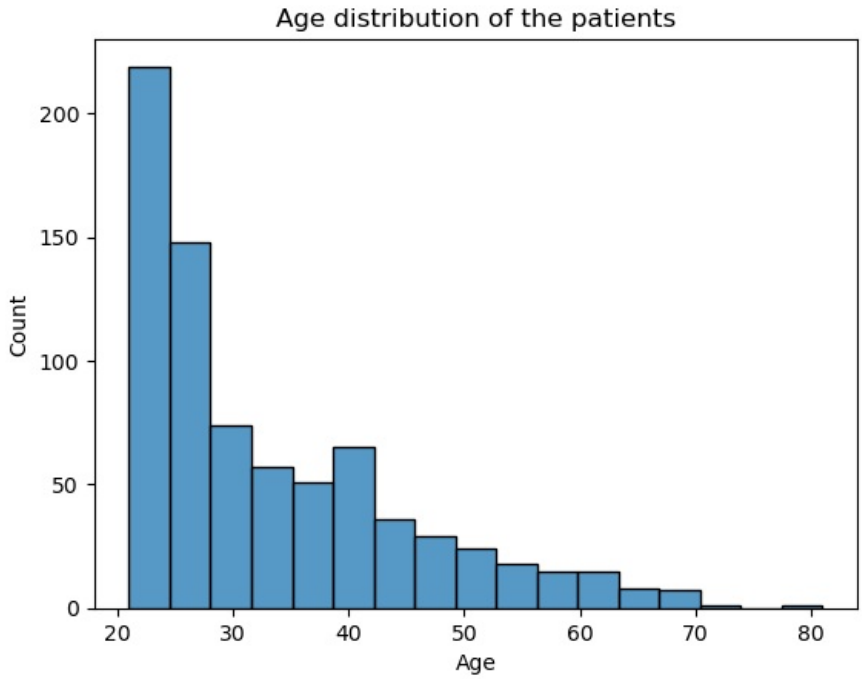
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	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 [384…
```
df[df.duplicated()]
```

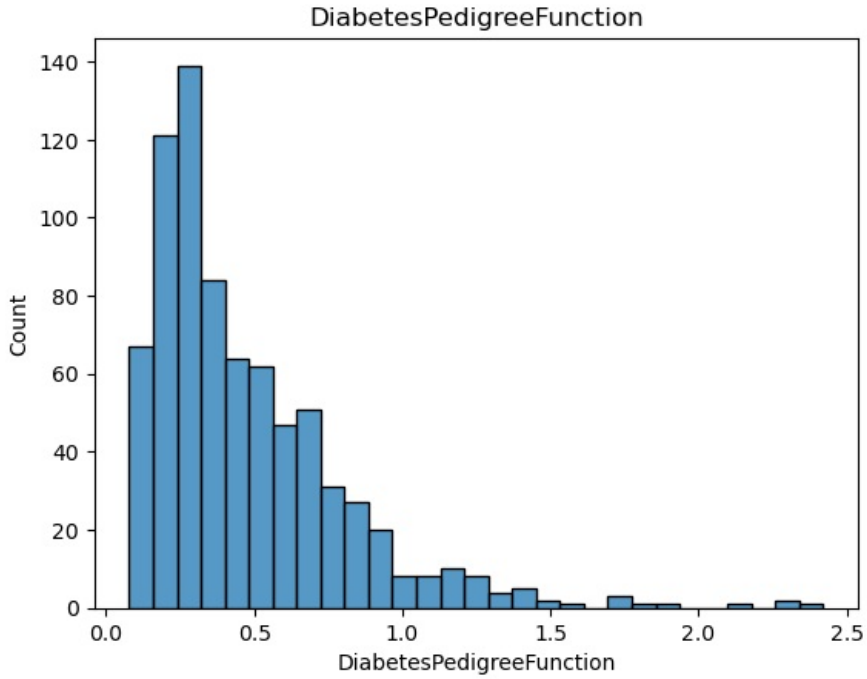
Out[384]:

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome

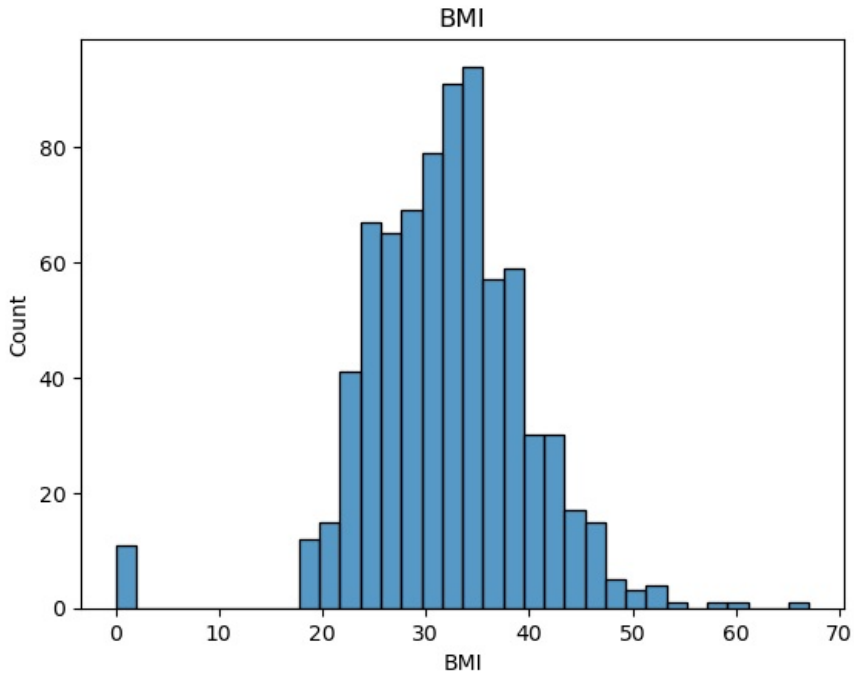
In [385…
```
sns.histplot(df.Age)
plt.title('Age distribution of the patients')
plt.show()
```



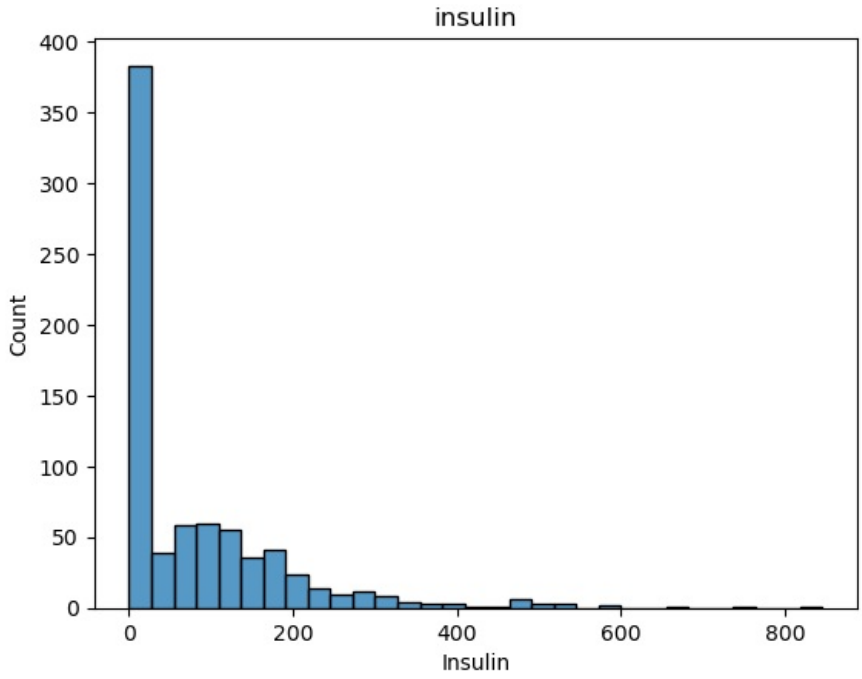
In [386…
```
sns.histplot(df.DiabetesPedigreeFunction)
plt.title('DiabetesPedigreeFunction')
plt.show()
```



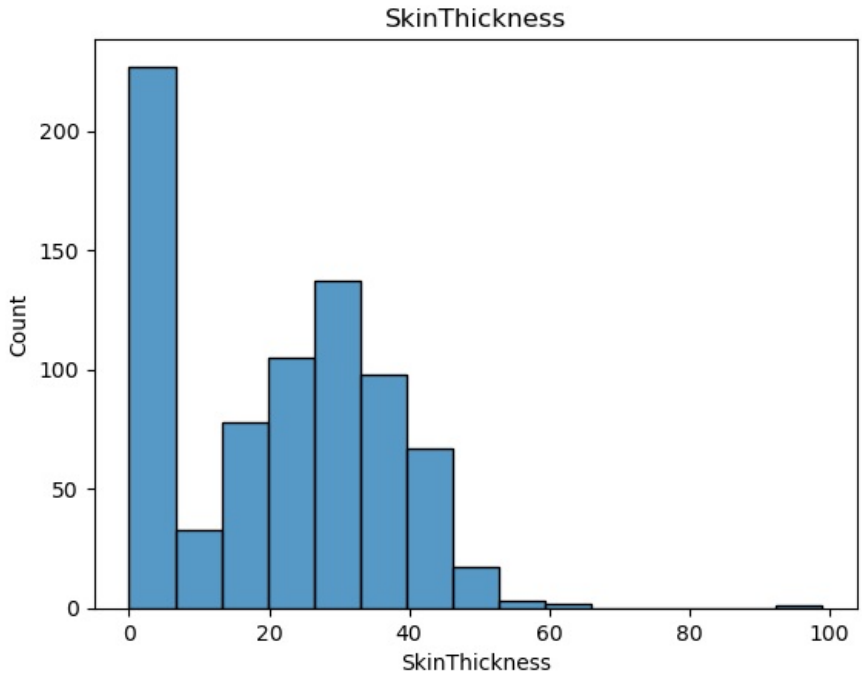
In [387…
```
sns.histplot(df.BMI)
plt.title('BMI')
plt.show()
```



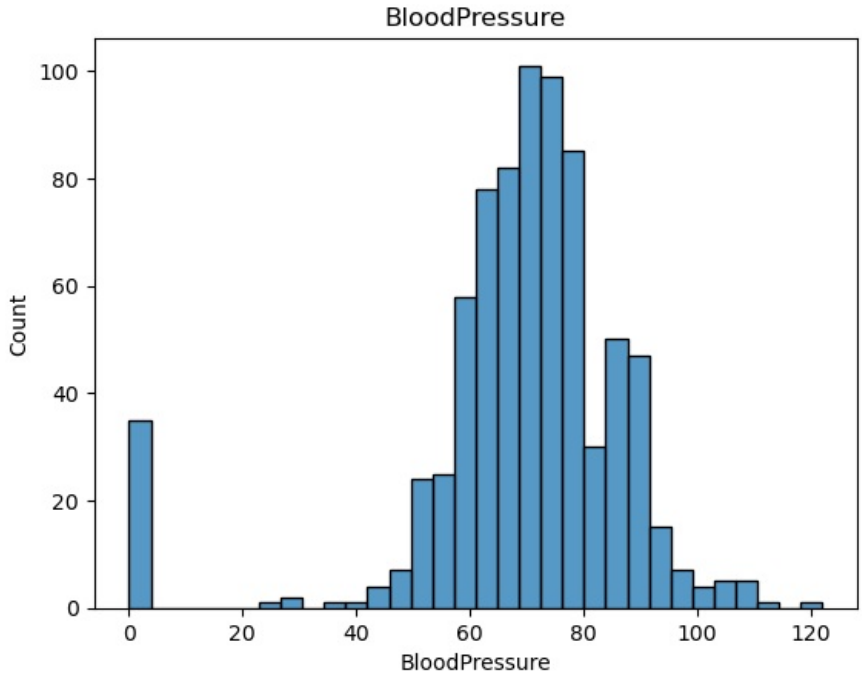
In [388…
```
sns.histplot(df.Insulin)
plt.title('insulin')
plt.show()
```



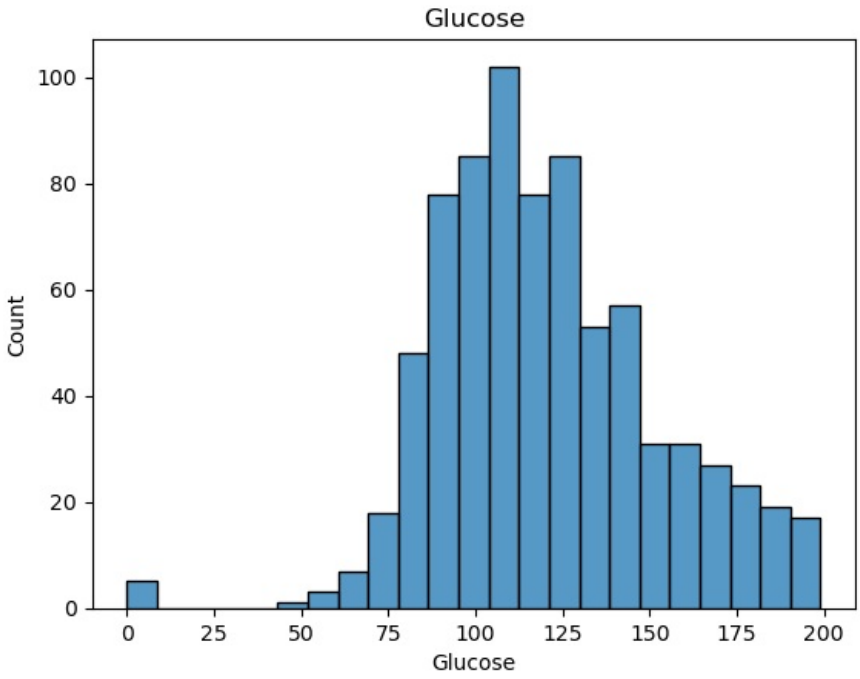
In [389…
```
sns.histplot(df.SkinThickness)
plt.title('SkinThickness')
plt.show()
```



In [390…
```
sns.histplot(df.BloodPressure)
plt.title('BloodPressure')
plt.show()
```

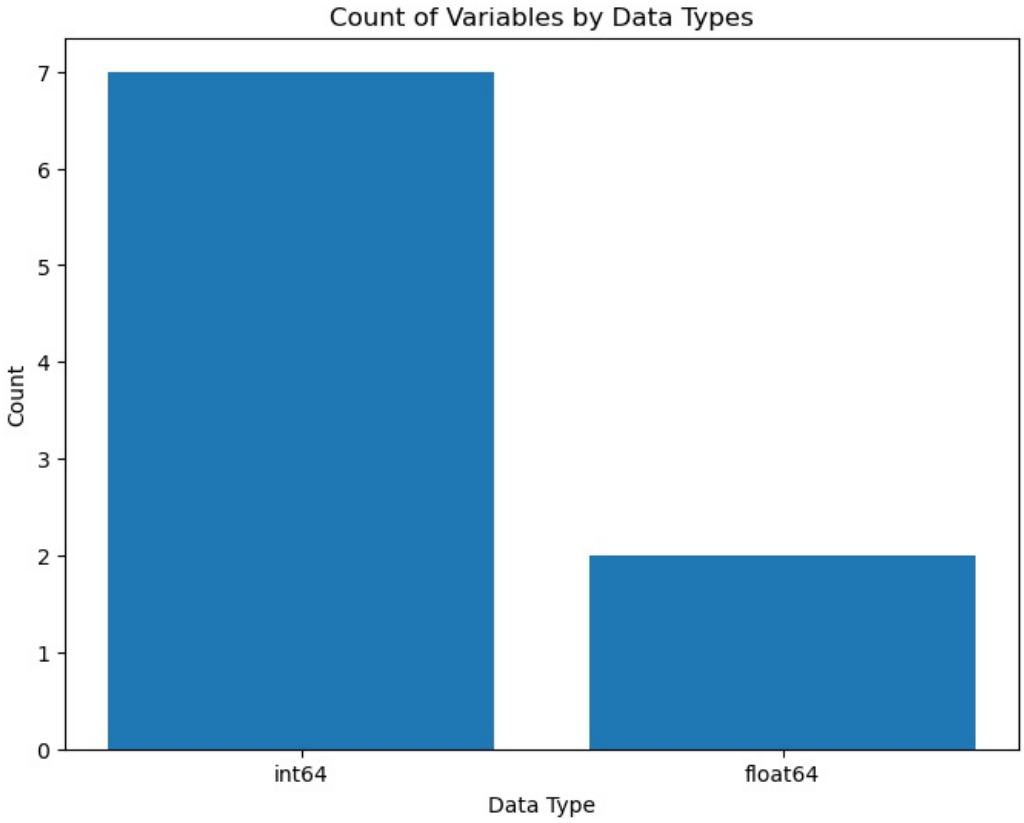


In [391…
```
sns.histplot(df.Glucose)
plt.title('Glucose')
plt.show()
```



In [392…
```
data_types_count = df.dtypes.value_counts()
```

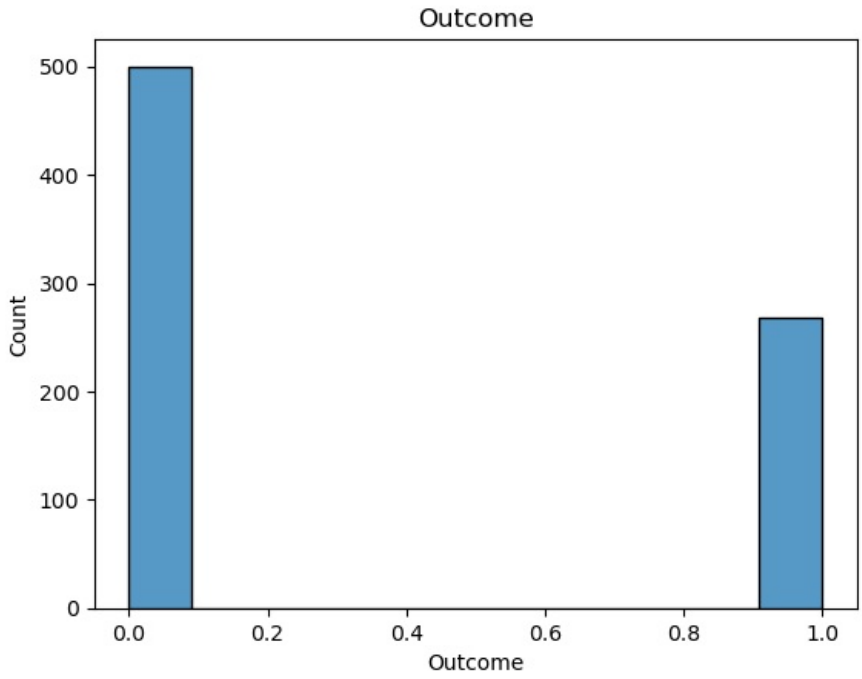
In [393…
```
plt.figure(figsize=(8, 6))
plt.bar(data_types_count.index.astype(str), data_types_count.values)
plt.xlabel('Data Type')
plt.ylabel('Count')
plt.title('Count of Variables by Data Types')
plt.show()
```



In [394…
```
df.Outcome.value_counts()
```

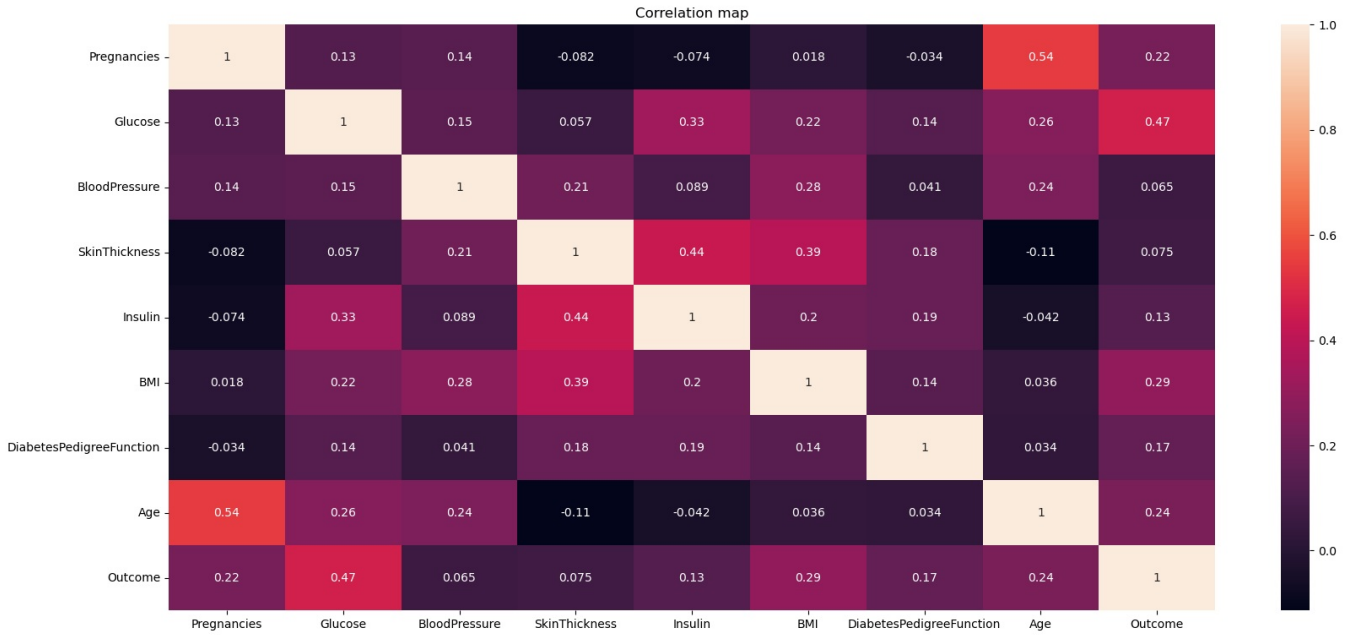
Out[394]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [395…
```
sns.histplot(df.Outcome)
plt.title('Outcome')
plt.show()
```



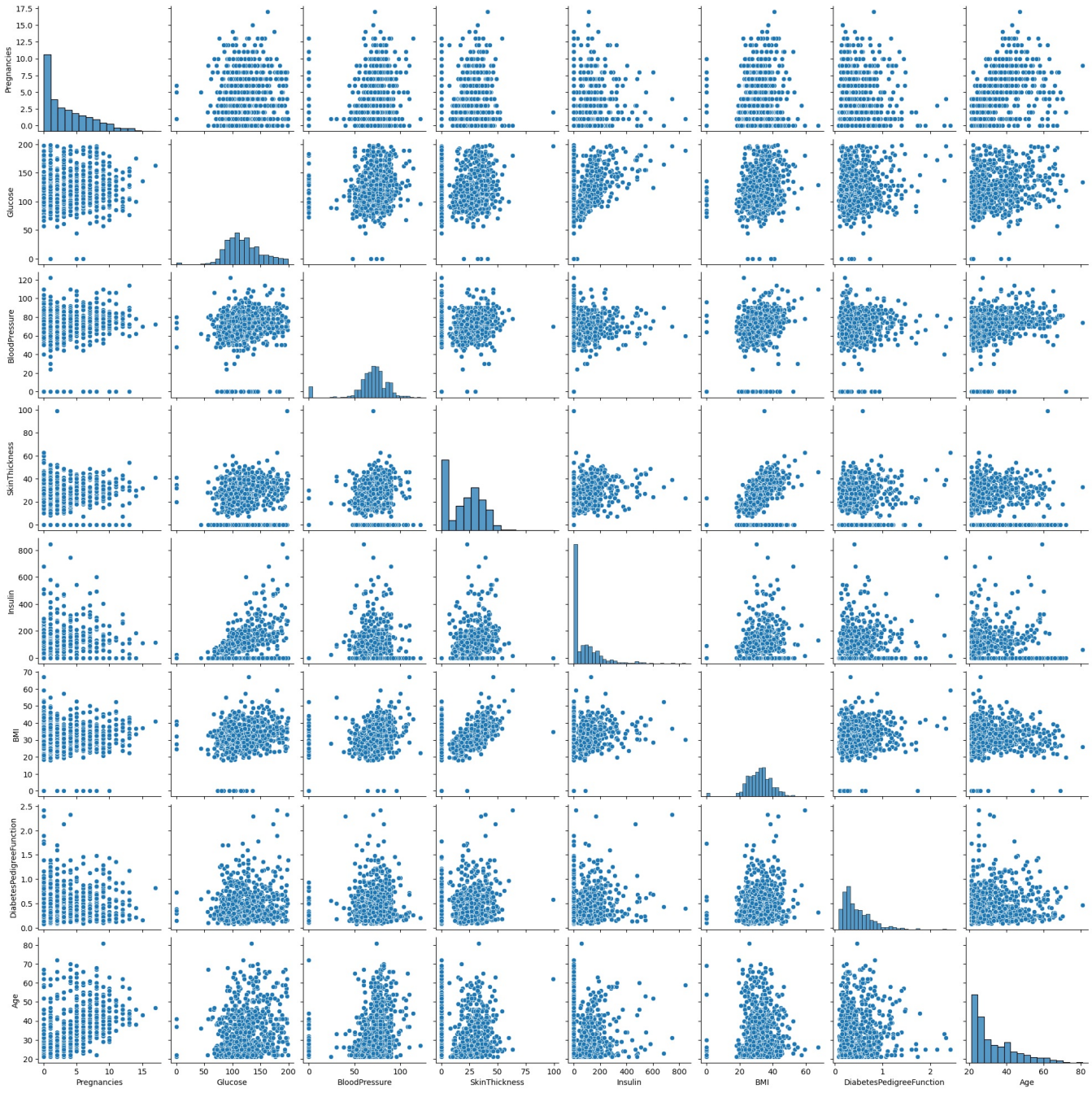
In [396…
```
plt.figure(figsize=(20,9));
sns.heatmap(df.corr(),annot=True);
plt.title("Correlation map")
```

Out[396]:
```
Text(0.5, 1.0, 'Correlation map')
```



In [397…
```
sns.pairplot(df[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunct
```

Out[397]:
```
<seaborn.axisgrid.PairGrid at 0x1b9aafe1fa0>
```

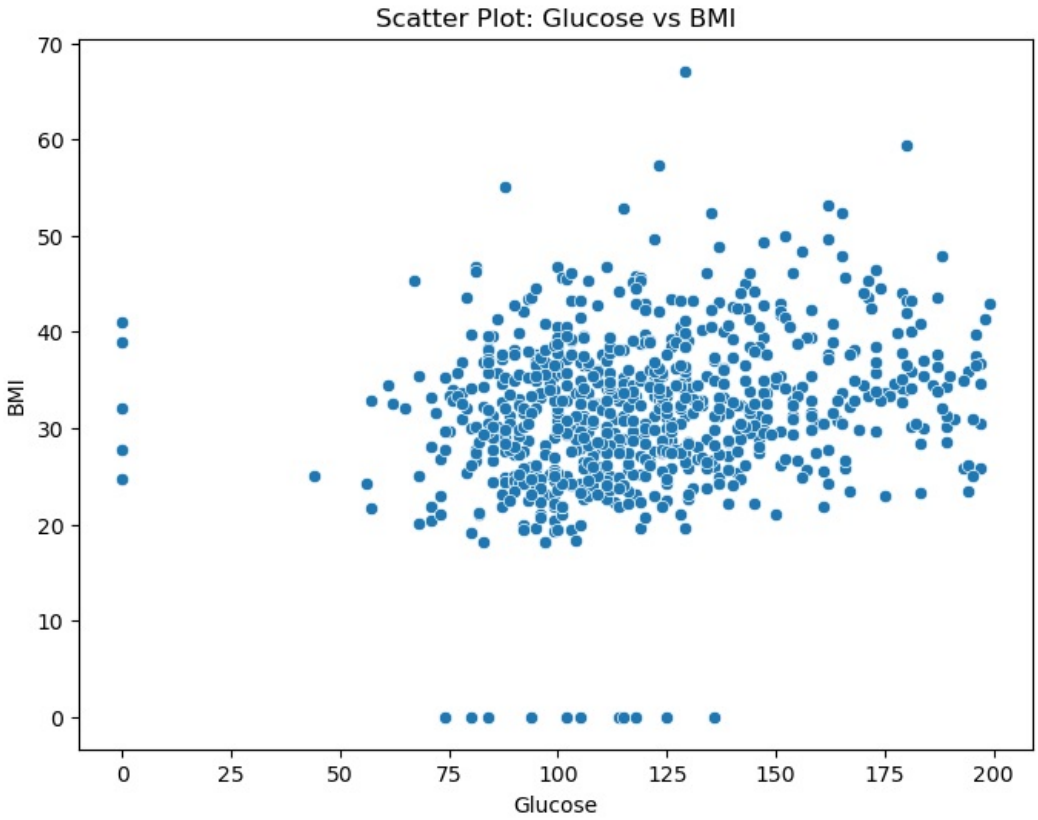


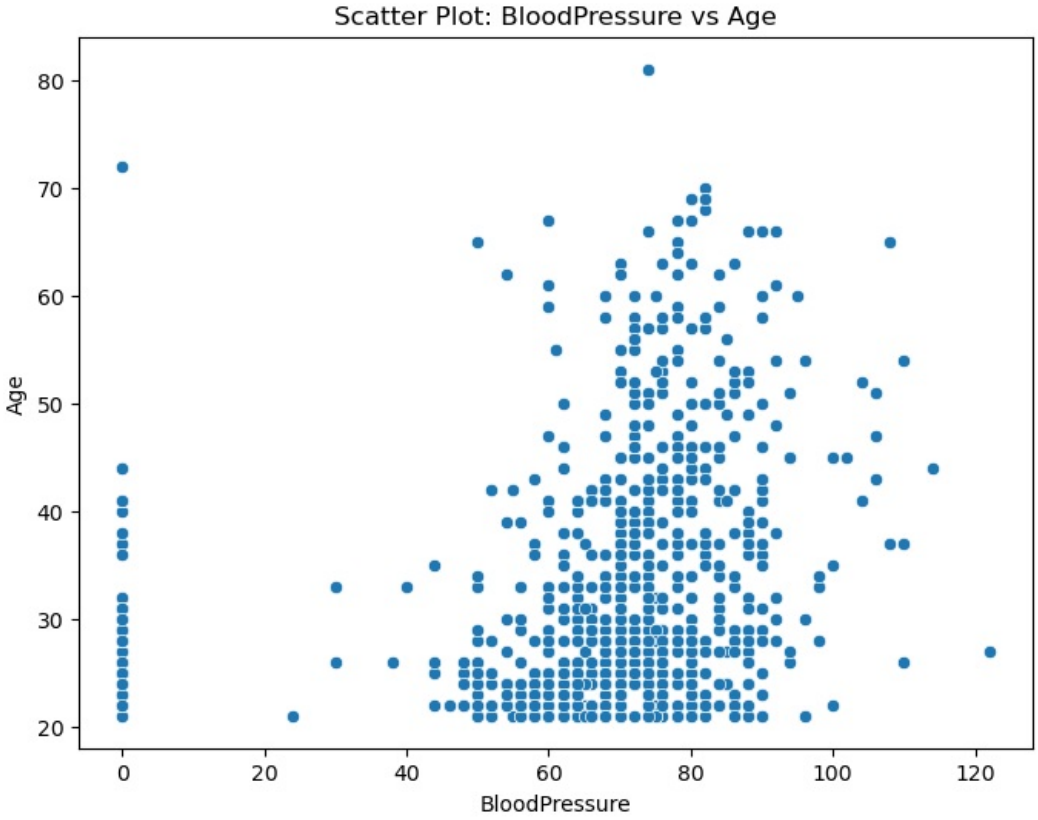
In [398…

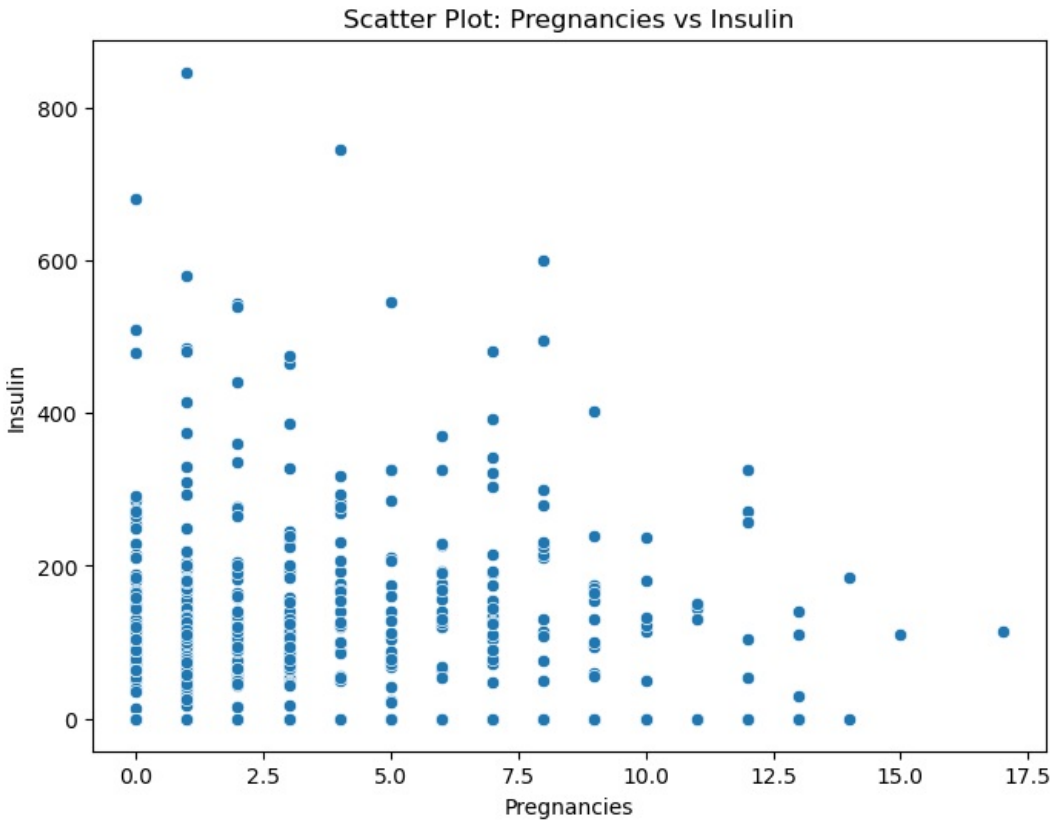
```
variable_pairs = [('Glucose', 'BMI'), ('BloodPressure', 'Age'), ('Pregnancies', 'Insulin'),('Glucose','Insulin'
```

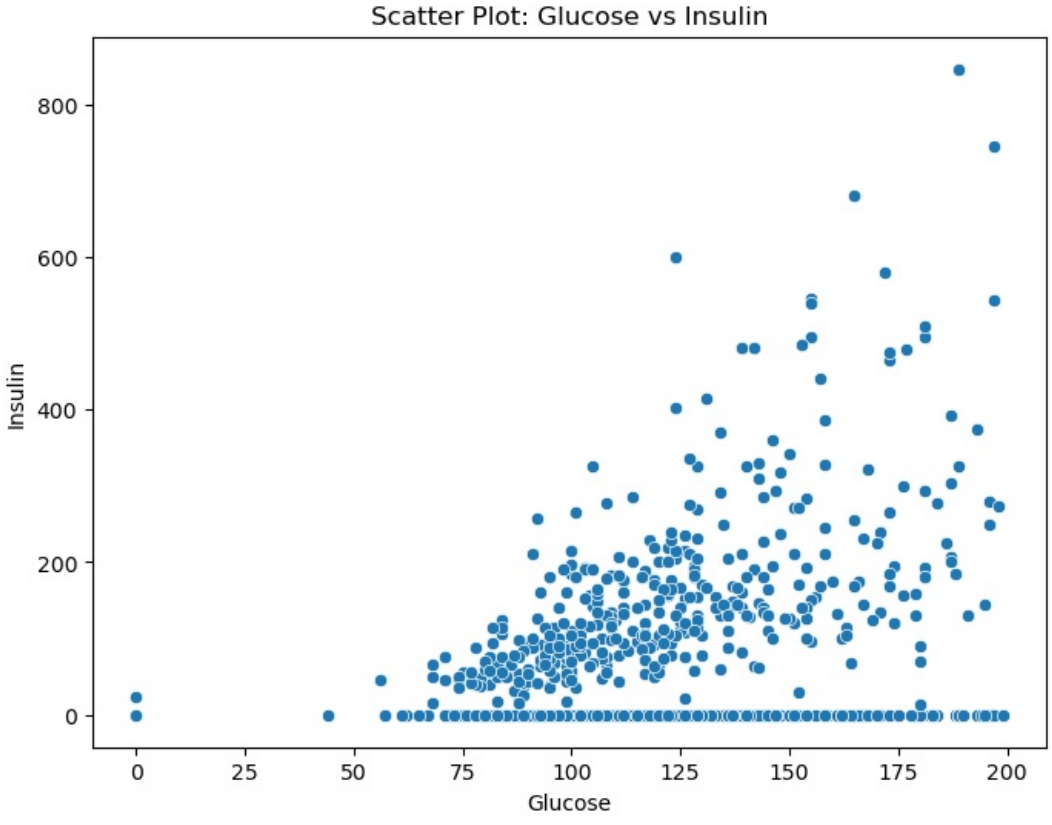
In [399…

```
for pair in variable_pairs:
    plt.figure(figsize=(8, 6))
    sns.scatterplot(data=df, x=pair[0], y=pair[1])
    plt.xlabel(pair[0])
    plt.ylabel(pair[1])
    plt.title(f'Scatter Plot: {pair[0]} vs {pair[1]}')
    plt.show()
```


Scatter Plot: Glucose vs BMI
BMI
Glucose
0
10
20
30
40
50
60
70
0
25
50
75
100
125
150
175
200


Scatter Plot: BloodPressure vs Age
Age
BloodPressure
20
30
40
50
60
70
80
0
20
40
60
80
100
120


Scatter Plot: Pregnancies vs Insulin
Insulin
Pregnancies
0
200
400
600
800
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Scatter Plot: Glucose vs Insulin
Insulin
Glucose
0
200
400
600
800
0
25
50
75
100
125
150
175
200

In [400…
```
df.head()
```

Out[400]:

	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 [401…
```
from sklearn.preprocessing import StandardScaler

scale = StandardScaler()
df[['Age','Pregnancies', 'Glucose','BloodPressure','SkinThickness','Insulin','BMI']]=scale.fit_transform(df[['A
```

In [402…
```
df.head()
```

Out[402]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.639947	0.848324	0.149641	0.907270	-0.692891	0.204013	0.627	1.425995	1
1	-0.844885	-1.123396	-0.160546	0.530902	-0.692891	-0.684422	0.351	-0.190672	0
2	1.233880	1.943724	-0.263941	-1.288212	-0.692891	-1.103255	0.672	-0.105584	1
3	-0.844885	-0.998208	-0.160546	0.154533	0.123302	-0.494043	0.167	-1.041549	0
4	-1.141852	0.504055	-1.504687	0.907270	0.765836	1.409746	2.288	-0.020496	1

In [403…
```
from sklearn.preprocessing import MinMaxScaler

scale = MinMaxScaler()
df[['Age','Pregnancies', 'Glucose','BloodPressure','SkinThickness','Insulin','BMI']]=scale.fit_transform(df[['A
```

In [404…
```
df.head()
```

Out[404]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.627	0.483333	1
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.351	0.166667	0
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.672	0.183333	1
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.167	0.000000	0
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	2.288	0.200000	1

In [405…
```
from sklearn.model_selection import train_test_split
```

In [406…
```
feature_cols = ['Age','Pregnancies', 'Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigre
X = df[feature_cols]
y = df.Outcome
```

In [407…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)
```

In [408…
```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[408]:
```
((614, 8), (154, 8), (614,), (154,))
```

In [409…
```
x = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

LogisticRegression

In [410…
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

In [411…
```
logreg.fit(X_train,y_train)
```

Out[411]:
```
LogisticRegression()
```

In [412…
```
y_pred = logreg.predict(X_test)
y_pred
```

Out[412]:
```
array([0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1],
      dtype=int64)
```

In [413…
```
for i in range(len(X_test)):
            print(logreg.predict_proba(X_test)[i])
```

```
[0.77398461 0.22601539]
[0.8560157 0.1439843]
[0.84580899 0.15419101]
[0.6440202 0.3559798]
[0.78291376 0.21708624]
[0.29622249 0.70377751]
[0.53661256 0.46338744]
[0.49513849 0.50486151]
[0.16623983 0.83376017]
[0.74786417 0.25213583]
[0.53636484 0.46363516]
[0.90061523 0.09938477]
[0.7307741 0.2692259]
[0.90033975 0.09966025]
[0.82334497 0.17665503]
[0.42899646 0.57100354]
[0.20257096 0.79742904]
[0.14435941 0.85564059]
[0.90688014 0.09311986]
[0.90544622 0.09455378]
[0.82547877 0.17452123]
[0.38427388 0.61572612]
[0.68015124 0.31984876]
[0.29687706 0.70312294]
[0.76569377 0.23430623]
[0.69889295 0.30110705]
[0.85860712 0.14139288]
[0.48170971 0.51829029]
[0.70774869 0.29225131]
[0.76494903 0.23505097]
[0.32244293 0.67755707]
[0.89657838 0.10342162]
[0.65162799 0.34837201]
[0.90297739 0.09702261]
[0.45979296 0.54020704]
[0.56358358 0.43641642]
[0.52079915 0.47920085]
[0.46308238 0.53691762]
[0.81862249 0.18137751]
[0.36729084 0.63270916]
[0.42918361 0.57081639]
[0.70831182 0.29168818]
[0.73225347 0.26774653]
[0.35343107 0.64656893]
[0.83422844 0.16577156]
[0.80067552 0.19932448]
[0.67396839 0.32603161]
[0.22553046 0.77446954]
[0.75366951 0.24633049]
[0.83179411 0.16820589]
[0.30560963 0.69439037]
[0.38150994 0.61849006]
[0.77654642 0.22345358]
[0.49611525 0.50388475]
[0.69888256 0.30111744]
[0.7898177 0.2101823]
[0.64252521 0.35747479]
[0.94179786 0.05820214]
[0.78633641 0.21366359]
[0.59693249 0.40306751]
[0.56541026 0.43458974]
[0.84125572 0.15874428]
[0.61873733 0.38126267]
[0.91471424 0.08528576]
[0.73971399 0.26028601]
[0.70305944 0.29694056]
[0.84146552 0.15853448]
[0.51039603 0.48960397]
[0.682323 0.317677]
[0.51629468 0.48370532]
[0.88635073 0.11364927]
[0.62665521 0.37334479]
[0.70748589 0.29251411]
[0.90794433 0.09205567]
[0.87065615 0.12934385]
[0.27637562 0.72362438]
[0.77610342 0.22389658]
```

```
[0.84872823 0.15127177]
[0.85384134 0.14615866]
[0.18871342 0.81128658]
[0.21366237 0.78633763]
[0.50873617 0.49126383]
[0.85077395 0.14922605]
[0.18346067 0.81653933]
[0.47097266 0.52902734]
[0.90426 0.09574]
[0.90006222 0.09993778]
[0.15730635 0.84269365]
[0.78789693 0.21210307]
[0.76688607 0.23311393]
[0.45745253 0.54254747]
[0.90775126 0.09224874]
[0.67958449 0.32041551]
[0.80892066 0.19107934]
[0.83462792 0.16537208]
[0.70758938 0.29241062]
[0.84232965 0.15767035]
[0.6157964 0.3842036]
[0.35838472 0.64161528]
[0.6727953 0.3272047]
[0.78476525 0.21523475]
[0.53008362 0.46991638]
[0.81480544 0.18519456]
[0.69418527 0.30581473]
[0.31821118 0.68178882]
[0.66960117 0.33039883]
[0.84572619 0.15427381]
[0.32550916 0.67449084]
[0.63178674 0.36821326]
[0.63782348 0.36217652]
[0.29347539 0.70652461]
[0.25135457 0.74864543]
[0.17230234 0.82769766]
[0.90438043 0.09561957]
[0.85959306 0.14040694]
[0.89610682 0.10389318]
[0.61967088 0.38032912]
[0.53681611 0.46318389]
[0.47075732 0.52924268]
[0.18621139 0.81378861]
[0.7872409 0.2127591]
[0.82768041 0.17231959]
[0.74327776 0.25672224]
[0.78406759 0.21593241]
[0.80164587 0.19835413]
[0.76357138 0.23642862]
[0.48670624 0.51329376]
[0.77048644 0.22951356]
[0.86017901 0.13982099]
[0.40712722 0.59287278]
[0.77672349 0.22327651]
[0.87786874 0.12213126]
[0.72710033 0.27289967]
[0.87763055 0.12236945]
[0.47883502 0.52116498]
[0.85426144 0.14573856]
[0.8166629 0.1833371]
[0.88434341 0.11565659]
[0.43701099 0.56298901]
[0.04158829 0.95841171]
[0.86785627 0.13214373]
[0.89537734 0.10462266]
[0.75884849 0.24115151]
[0.63959633 0.36040367]
[0.48748897 0.51251103]
[0.38767591 0.61232409]
[0.42090337 0.57909663]
[0.64905112 0.35094888]
[0.83996638 0.16003362]
[0.56365438 0.43634562]
[0.8737704 0.1262296]
[0.6007933 0.3992067]
[0.37594119 0.62405881]
[0.34809374 0.65190626]
```

In [414…
```
from sklearn import metrics
```

In [415…
```
metrics.accuracy_score(y_test, y_pred)
```

Out[415]:
```
0.8051948051948052
```

In [416…
```
metrics.confusion_matrix(y_test, y_pred)
```

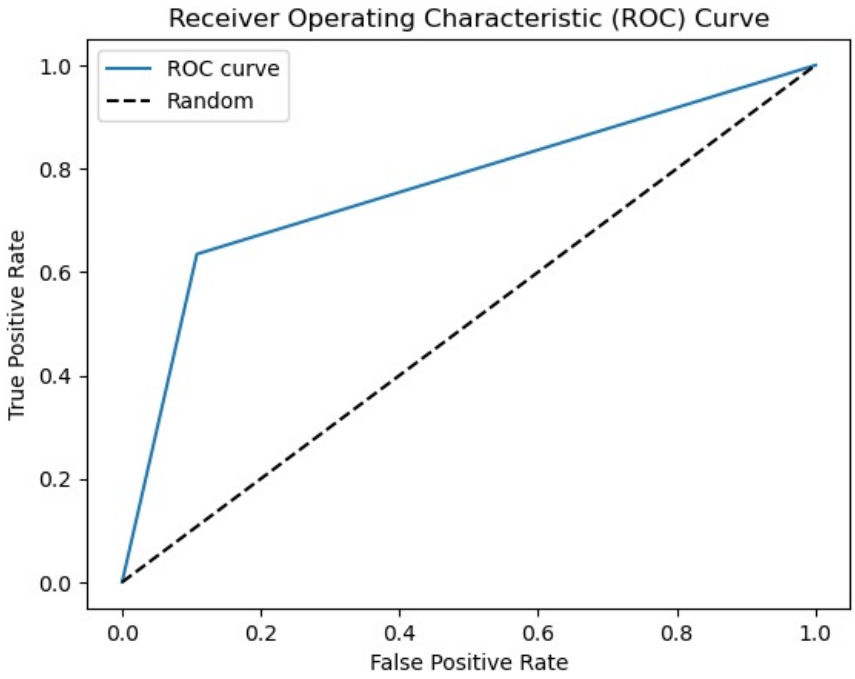
Out[416]:
```
array([[91, 11],
       [19, 33]], dtype=int64)
```

In [417…
```
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86       102
           1       0.75      0.63      0.69        52

    accuracy                           0.81       154
   macro avg       0.79      0.76      0.77       154
weighted avg       0.80      0.81      0.80       154
```

In [330…
```
auc = roc_auc_score(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)

print("AUC:", auc)

# Plot the ROC curve
plt.plot(fpr, tpr, label='ROC curve')
plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```

```
AUC: 0.7633861236802413
```



DecisionTreeClassifier

In [331…
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
```

In [332…
```
def mymodel(model):
    model.fit(X_train,y_train)
    ypred=model.predict(X_test)
    print(classification_report(y_test,y_pred))
    return model
```

In [333…
```
 print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86       102
           1       0.75      0.63      0.69        52

    accuracy                           0.81       154
   macro avg       0.79      0.76      0.77       154
weighted avg       0.80      0.81      0.80       154
```

In [334…
```
df1=DecisionTreeClassifier(max_depth=10)
mymodel(df1)
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86       102
           1       0.75      0.63      0.69        52

    accuracy                           0.81       154
   macro avg       0.79      0.76      0.77       154
weighted avg       0.80      0.81      0.80       154
```

Out[334]:
```
DecisionTreeClassifier(max_depth=10)
```

In [335…
```
for i in range(1,50):
    dt2=DecisionTreeClassifier(max_depth=i)
    dt2.fit(X_train,y_train)
    ypred=dt2.predict(X_test)
    print(f"{i}: {accuracy_score(y_test,y_pred)}")
```

```
1: 0.8051948051948052
2: 0.8051948051948052
3: 0.8051948051948052
4: 0.8051948051948052
5: 0.8051948051948052
6: 0.8051948051948052
7: 0.8051948051948052
8: 0.8051948051948052
9: 0.8051948051948052
10: 0.8051948051948052
11: 0.8051948051948052
12: 0.8051948051948052
13: 0.8051948051948052
14: 0.8051948051948052
15: 0.8051948051948052
16: 0.8051948051948052
17: 0.8051948051948052
18: 0.8051948051948052
19: 0.8051948051948052
20: 0.8051948051948052
21: 0.8051948051948052
22: 0.8051948051948052
23: 0.8051948051948052
24: 0.8051948051948052
25: 0.8051948051948052
26: 0.8051948051948052
27: 0.8051948051948052
28: 0.8051948051948052
29: 0.8051948051948052
30: 0.8051948051948052
31: 0.8051948051948052
32: 0.8051948051948052
33: 0.8051948051948052
34: 0.8051948051948052
35: 0.8051948051948052
36: 0.8051948051948052
37: 0.8051948051948052
38: 0.8051948051948052
39: 0.8051948051948052
40: 0.8051948051948052
41: 0.8051948051948052
42: 0.8051948051948052
43: 0.8051948051948052
44: 0.8051948051948052
45: 0.8051948051948052
46: 0.8051948051948052
47: 0.8051948051948052
48: 0.8051948051948052
49: 0.8051948051948052
```

In [336…
```
df3=DecisionTreeClassifier(max_depth=5)
mymodel(df3)
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86       102
           1       0.75      0.63      0.69        52

    accuracy                           0.81       154
   macro avg       0.79      0.76      0.77       154
weighted avg       0.80      0.81      0.80       154
```

Out[336]:
```
DecisionTreeClassifier(max_depth=5)
```

In [337…
```
df4=DecisionTreeClassifier(min_samples_leaf=20)  #The minimum number of samples required to be at a leaf node.
mymodel(df4)
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86       102
           1       0.75      0.63      0.69        52

    accuracy                           0.81       154
   macro avg       0.79      0.76      0.77       154
weighted avg       0.80      0.81      0.80       154
```

Out[337]:
```
DecisionTreeClassifier(min_samples_leaf=20)
```

In [338…
```
for i in range(1,75):
    dt2=DecisionTreeClassifier(max_depth=i)
    dt2.fit(X_train,y_train)
    ypred=dt2.predict(X_test)
    print(f"{i}: {accuracy_score(y_test,y_pred)}")
```

```
1: 0.8051948051948052
2: 0.8051948051948052
3: 0.8051948051948052
4: 0.8051948051948052
5: 0.8051948051948052
6: 0.8051948051948052
7: 0.8051948051948052
8: 0.8051948051948052
9: 0.8051948051948052
10: 0.8051948051948052
11: 0.8051948051948052
12: 0.8051948051948052
13: 0.8051948051948052
14: 0.8051948051948052
15: 0.8051948051948052
16: 0.8051948051948052
17: 0.8051948051948052
18: 0.8051948051948052
19: 0.8051948051948052
20: 0.8051948051948052
21: 0.8051948051948052
22: 0.8051948051948052
23: 0.8051948051948052
24: 0.8051948051948052
25: 0.8051948051948052
26: 0.8051948051948052
27: 0.8051948051948052
28: 0.8051948051948052
29: 0.8051948051948052
30: 0.8051948051948052
31: 0.8051948051948052
32: 0.8051948051948052
33: 0.8051948051948052
34: 0.8051948051948052
35: 0.8051948051948052
36: 0.8051948051948052
37: 0.8051948051948052
38: 0.8051948051948052
39: 0.8051948051948052
40: 0.8051948051948052
41: 0.8051948051948052
42: 0.8051948051948052
43: 0.8051948051948052
44: 0.8051948051948052
45: 0.8051948051948052
46: 0.8051948051948052
47: 0.8051948051948052
48: 0.8051948051948052
49: 0.8051948051948052
50: 0.8051948051948052
51: 0.8051948051948052
52: 0.8051948051948052
53: 0.8051948051948052
54: 0.8051948051948052
55: 0.8051948051948052
56: 0.8051948051948052
57: 0.8051948051948052
58: 0.8051948051948052
59: 0.8051948051948052
60: 0.8051948051948052
61: 0.8051948051948052
62: 0.8051948051948052
63: 0.8051948051948052
64: 0.8051948051948052
65: 0.8051948051948052
66: 0.8051948051948052
67: 0.8051948051948052
68: 0.8051948051948052
69: 0.8051948051948052
70: 0.8051948051948052
71: 0.8051948051948052
72: 0.8051948051948052
73: 0.8051948051948052
74: 0.8051948051948052
```

In [339…

```
df9=DecisionTreeClassifier(criterion="entropy",max_depth=5,min_samples_leaf=20)
mymodel(df9)
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86       102
           1       0.75      0.63      0.69        52

    accuracy                           0.81       154
   macro avg       0.79      0.76      0.77       154
weighted avg       0.80      0.81      0.80       154
```

Out[339]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=5, min_samples_leaf=20)
```

In [340…]
```
from sklearn.model_selection import GridSearchCV

params = {'criterion' : ['gini', 'entropy'],
          'max_depth' : [ 3, 4, 5, 7],
          'min_samples_leaf' : [10, 20, 50,100,150],
         }

grid_search = GridSearchCV(dt, param_grid= params)
```

In [341…]
```
grid_search.fit(X_train, y_train)
```

Out[341]:
```
GridSearchCV(estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 4, 5, 7],
                         'min_samples_leaf': [10, 20, 50, 100, 150]})
```

In [342…]
```
grid_search.best_params_
```

Out[342]:
```
{'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 20}
```

In [343…]
```
my_best_preds = grid_search.predict(X_test)
```

In [344…]
```
accuracy_score(y_test, my_best_preds)
```

Out[344]:
```
0.7597402597402597
```

In [345…]
```
print(classification_report(y_test, my_best_preds))
```

```
              precision    recall  f1-score   support

           0       0.80      0.85      0.82       102
           1       0.67      0.58      0.62        52

    accuracy                           0.76       154
   macro avg       0.73      0.71      0.72       154
weighted avg       0.75      0.76      0.76       154
```

RandomForestClassifier

In [346…]
```
from sklearn.ensemble import RandomForestClassifier
my_rf_classifier = RandomForestClassifier()
```

In [347…]
```
my_rf_classifier.fit(X_train, y_train)
```

Out[347]:
```
RandomForestClassifier()
```

In [348…]
```
my_predictions = my_rf_classifier.predict(X_test)
```

In [349…]
```
print(accuracy_score(y_test, my_predictions))
```

```
0.7727272727272727
```

In [350…]
```
print(classification_report(y_test, my_predictions))
```

```
              precision    recall  f1-score   support

           0       0.82      0.84      0.83       102
           1       0.67      0.63      0.65        52

    accuracy                           0.77       154
   macro avg       0.75      0.74      0.74       154
weighted avg       0.77      0.77      0.77       154
```

#Support Vector Machine

In [351…]
```
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.metrics import classification_report,confusion_matrix
```

In [352…]
```
svc_model=SVC()
svc_model.fit(X_train,y_train)
y_pred=svc_model.predict(X_test)
```

In [353…]
```
print('Accuracy Score:')
print(metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy Score:
0.7792207792207793
```

In [354…]
```
print(confusion_matrix(y_test,y_pred))
```

```
[[87 15]
 [19 33]]
```

In [355…
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.85      0.84       102
           1       0.69      0.63      0.66        52

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

VotingClassifier

In [356…
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
```

In [357…
```
df_clf = DecisionTreeClassifier()
log_clf = LogisticRegression()
svm_clf = SVC()
```

In [358…
```
voting_clf = VotingClassifier(estimators=[('lr', log_clf), ('df', df_clf), ('svc', svm_clf)])
```

In [359…
```
voting_clf.fit(X_train, y_train)
```

Out[359]:
```
VotingClassifier(estimators=[('lr', LogisticRegression()),
                             ('df', DecisionTreeClassifier()), ('svc', SVC())])
```

In [360…
```
from sklearn.metrics import accuracy_score
```

In [361…
```
for clf in (log_clf, df_clf, svm_clf, voting_clf):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
```

```
LogisticRegression 0.8051948051948052
DecisionTreeClassifier 0.7207792207792207
SVC 0.7792207792207793
VotingClassifier 0.7857142857142857
```

BaggingClassifier

In [362…
```
from sklearn.ensemble import BaggingClassifier
```

In [363…
```
bag_clf = BaggingClassifier(log_clf,n_estimators=10)
```

In [364…
```
bag_clf.fit(X_train,y_train)
```

Out[364]:
```
BaggingClassifier(base_estimator=LogisticRegression())
```

In [365…
```
y_pred = bag_clf.predict(X_test)
accuracy_score(y_pred, y_test)
```

Out[365]:
```
0.7987012987012987
```

AdaBoostClassifier

In [366…
```
from sklearn.ensemble import AdaBoostClassifier

ada_clf = AdaBoostClassifier(DecisionTreeClassifier(), n_estimators= 100)
```

In [367…
```
ada_clf.fit(X_train, y_train)
```

Out[367]:
```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=100)
```

In [368…
```
y_pred = ada_clf.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[368]:
```
0.7012987012987013
```

In [369…
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.74      0.77       102
           1       0.55      0.63      0.59        52

    accuracy                           0.70       154
   macro avg       0.67      0.68      0.68       154
weighted avg       0.71      0.70      0.71       154
```

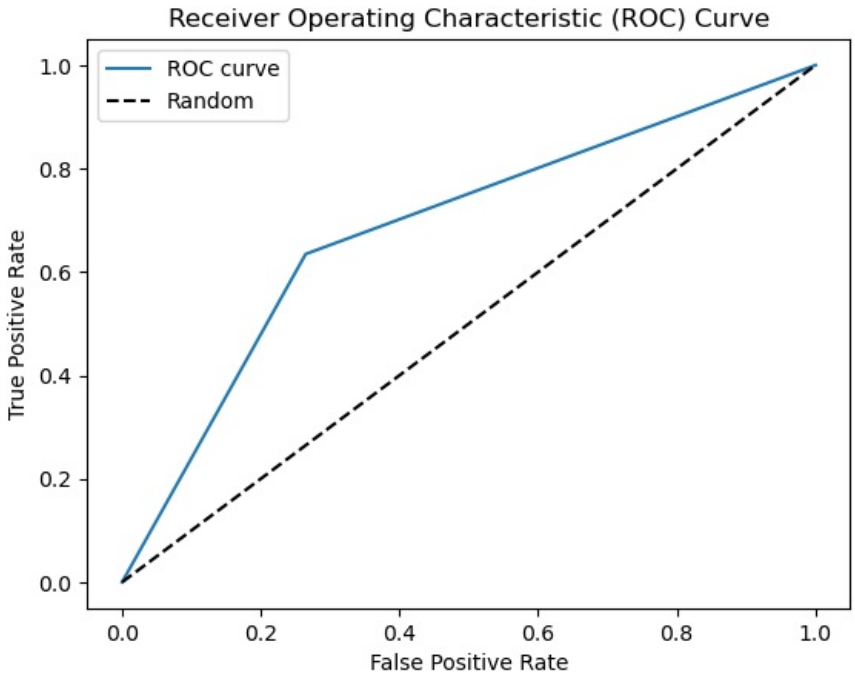
In [370…
```
from sklearn.metrics import classification_report, roc_auc_score, roc_curve
```

In [371…
```
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.80      0.74      0.77       102
           1       0.55      0.63      0.59        52

    accuracy                           0.70       154
   macro avg       0.67      0.68      0.68       154
weighted avg       0.71      0.70      0.71       154
```

In [372…
```
auc = roc_auc_score(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)

print("AUC:", auc)

# Plot the ROC curve
plt.plot(fpr, tpr, label='ROC curve')
plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```

```
AUC: 0.6849547511312218
```



In [ ]:

Tableau Public - health

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Standard Show Me

Data Analytics

health care diabetes (he...

Search

Tables

- Age (bin)
- Outcome (bin)
- Outcome (group)1
- *Measure Names*
- Age
- Blood Pressure
- BMI
- Diabetes Pedigree Function
- Glucose
- Insulin
- Outcome
- Pregnancies
- Skin Thickness
- *health care diabetes (Cou...*
- *Measure Values*

Pages

Filters

Marks

Pie

Color Size Label

Detail Tooltip Angle

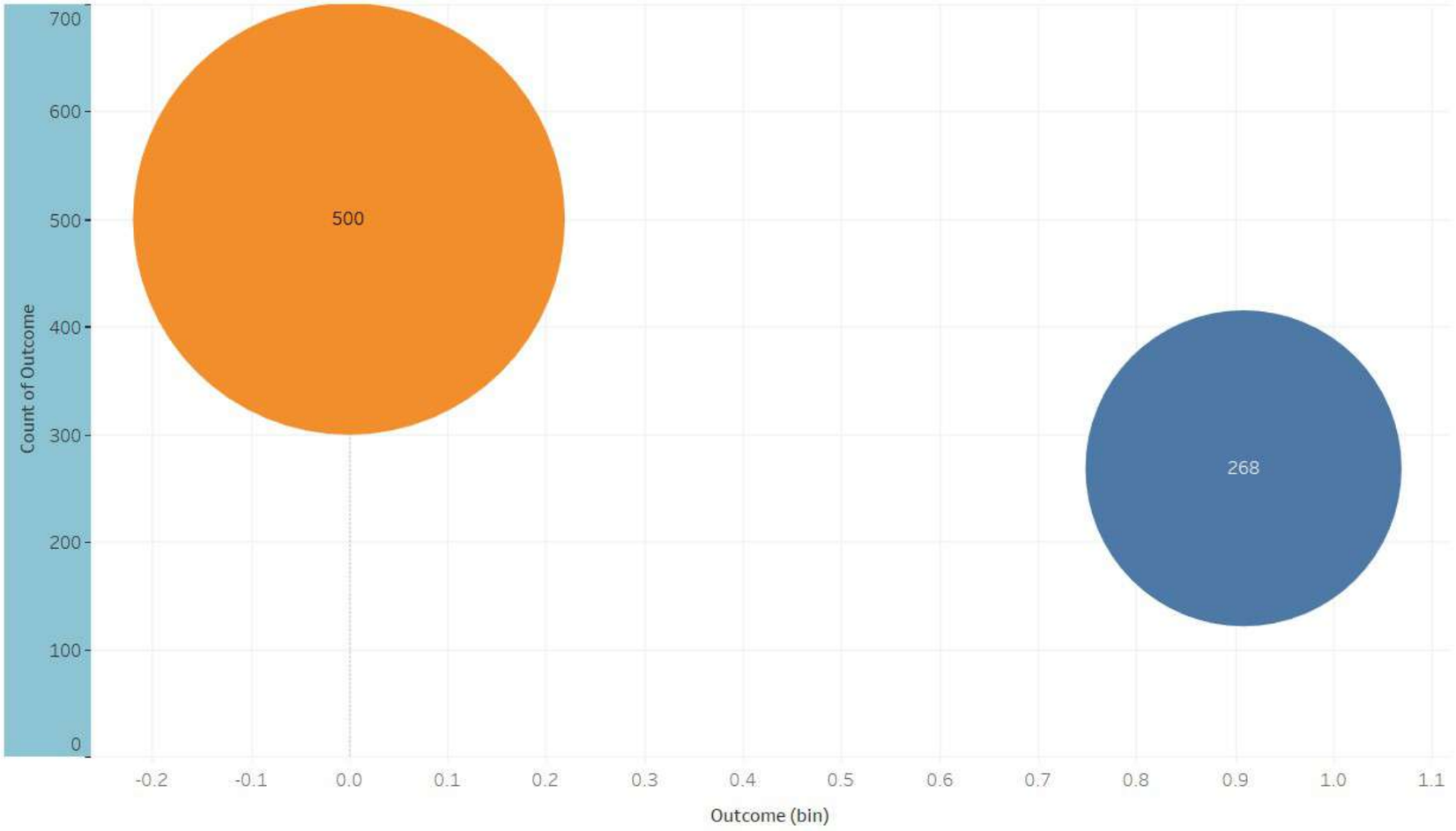
Outcome (gro..

CNT(Outcome)

Columns: Outcome (bin)

Rows: CNT(Outcome)

Pie chart to describe the diabetic or non-diabetic population



Outcome (group)1

Diabetes

Non-Diabetes

CNT(Outcome)

268
300
350
400
450
500

Data Source | Sheet 1 | Sheet 2 | Sheet 3 | Sheet 4 | Sheet 5 | Sheet 6 | Sheet 7 | Sheet 8 | Sheet 9 | Sheet 10 | Sheet 11

2 marks 1 row by 1 column SUM of CNT(Outcome): 768

Tableau Public - health

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Standard | Show Me

Data | Analytics

health care diabetes (he...

Search

Tables

- Age (bin)
- Outcome (bin)
- Outcome (group)1
- *Measure Names*
- Age
- Blood Pressure
- BMI
- Diabetes Pedigree Function
- Glucose
- Insulin
- Outcome
- Pregnancies
- Skin Thickness
- *health care diabetes (Cou...*
- *Measure Values*

Pages

Filters

Marks

Bar

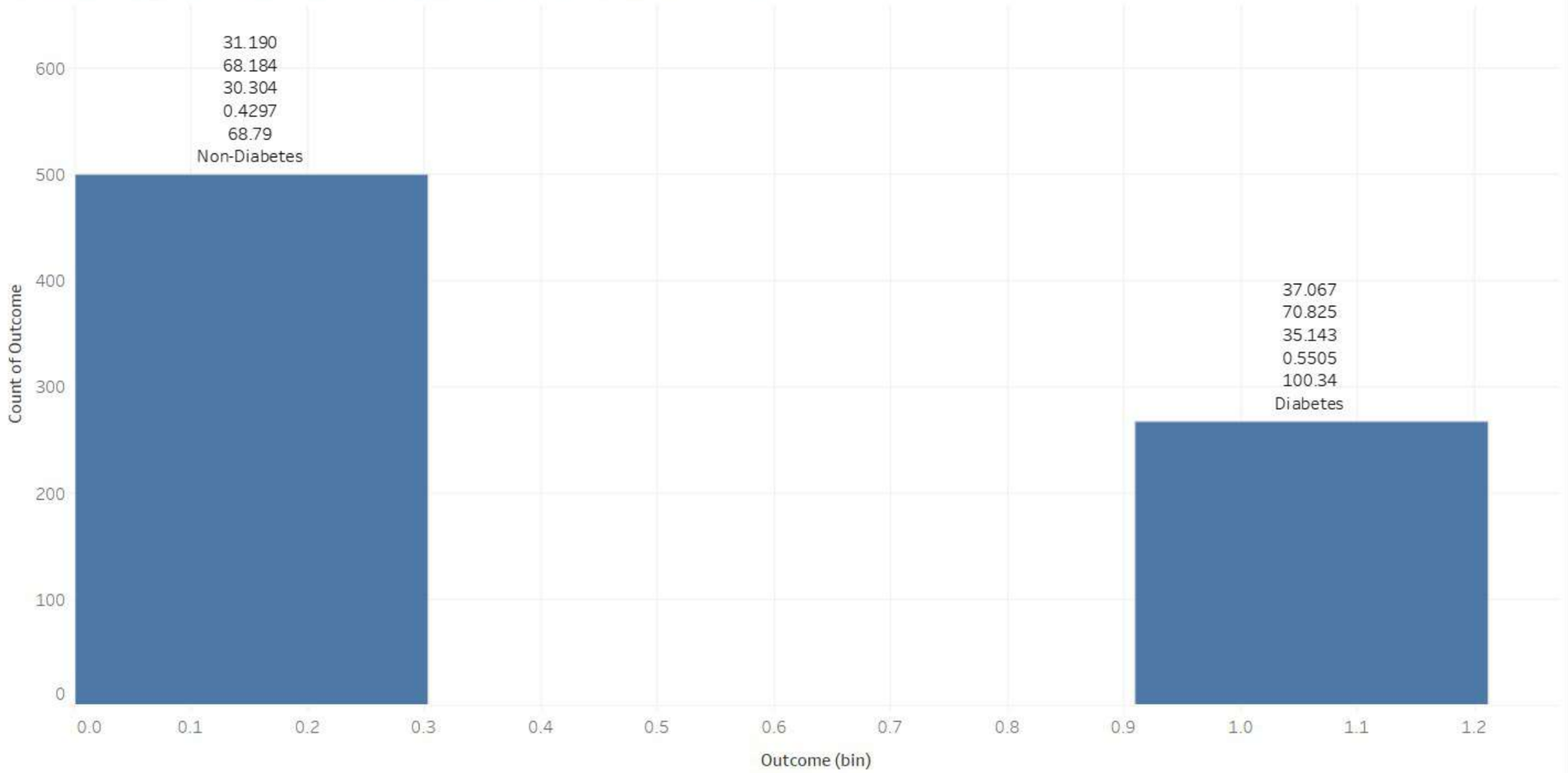
Color | Size | Label | Detail | Tooltip

AVG(Age)
AVG(Blood Pr..
AVG(BMI)
AVG(Diabetes ..
AVG(Insulin)
Outcome (gro..

Columns: Outcome (bin)

Rows: CNT(Outcome)

Histogram or frequency charts to analyze the distribution of the data



Data Source | Sheet 1 | Sheet 2 | Sheet 3 | Sheet 4 | Sheet 5 | Sheet 6 | Sheet 7 | Sheet 8 | Sheet 9 | Sheet 10 | Sheet 11

2 marks 1 row by 1 column SUM of CNT(Outcome): 768

Tableau Public - health

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Standard | Show Me

Data | Analytics

health care diabetes (he...

Search

Tables

- Age (bin)
- Outcome (bin)
- Outcome (group)1
- *Measure Names*
- Age
- Blood Pressure
- BMI
- Diabetes Pedigree Function
- Glucose
- Insulin
- Outcome
- Pregnancies
- Skin Thickness
- *health care diabetes (Cou...*
- *Measure Values*

Pages

Filters

Marks

Automatic

Color | Size | Label | Detail | Tooltip

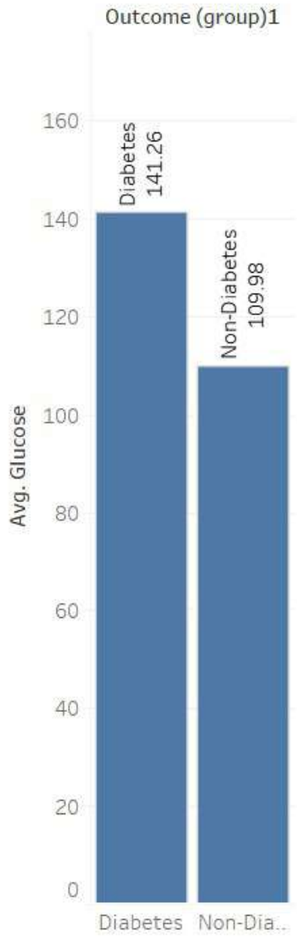
Outcome (gro..

AVG(Glucose)

Columns: Outcome (group)1

Rows: AVG(Glucose)

Sheet 3



Data Source | Sheet 1 | Sheet 2 | Sheet 3 | Sheet 4 | Sheet 5 | Sheet 6 | Sheet 7 | Sheet 8 | Sheet 9 | Sheet 10 | Sheet 11

marks 1 row by 2 columns SUM of AVG(Glucose): 251.24

Tableau Public - health

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Standard | Show Me

Data | Analytics

health care diabetes (he...

Search

Tables

- Age (bin)
- Outcome (bin)
- Outcome (group)1
- *Measure Names*
- Age
- Blood Pressure
- BMI
- Diabetes Pedigree Function
- Glucose
- Insulin
- Outcome
- Pregnancies
- Skin Thickness
- *health care diabetes (Cou...*
- *Measure Values*

Pages

Filters

Age (bin)

Marks

Circle

Color | Size | Label | Detail | Tooltip

AVG(Glucose)

CNT(Age)

Age (bin)

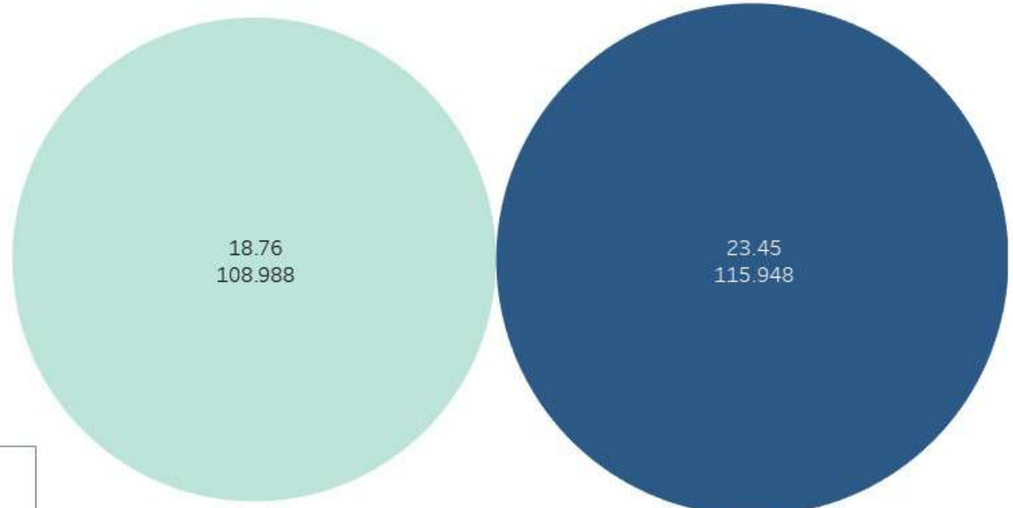
AVG(Glucose)

Title

Age(20-25) ,Avg Glucose Level Graph using Bubble Chart.

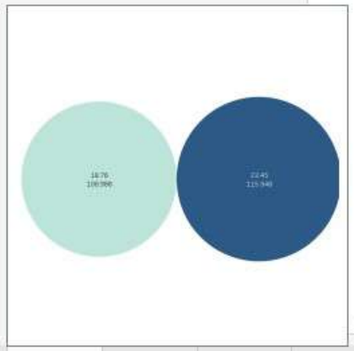
Columns

Rows



AVG(Glucose)

108.988 115.948



Data Source | Sheet 1 | Sheet 2 | Sheet 3 | Sheet 4 | **Sheet 5** | Sheet 6 | Sheet 7 | Sheet 8 | Sheet 9 | Sheet 10 | Sheet 11

2 marks 1 row by 1 column SUM of AVG(Glucose): 224.937

Tableau Public - health

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Standard Show Me

Data Analytics

health care diabetes (he...

Search

Tables

- Age (bin)
- Outcome (bin)
- Outcome (group)1
- *Measure Names*
- Age
- Blood Pressure
- BMI
- Diabetes Pedigree Function
- Glucose
- Insulin
- Outcome
- Pregnancies
- Skin Thickness
- *health care diabetes (Cou...*
- *Measure Values*

Pages

Columns

Rows

Filters

Age (bin)

Marks

Circle

Color Size Label

Detail Tooltip

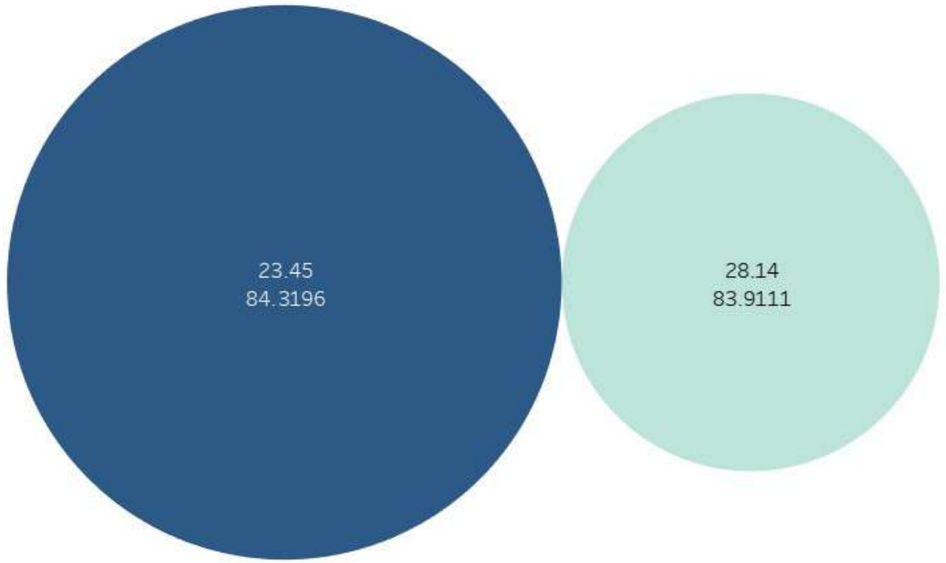
AVG(Insulin)

CNT(Age)

Age (bin)

AVG(Insulin)

Age(25-30) ,Avg Insulin Level Graph using Bubble Chart.



AVG(Insulin)

83.9111 84.3196

Data Source | Sheet 1 | Sheet 2 | Sheet 3 | Sheet 4 | Sheet 5 | **Sheet 6** | Sheet 7 | Sheet 8 | Sheet 9 | Sheet 10 | Sheet 11

2 marks 1 row by 1 column SUM of AVG(Insulin): 168.2307

Tableau Public - health

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Standard Show Me

Data Analytics

health care diabetes (he...

Search

Tables

- Age (bin)
- Outcome (bin)
- Outcome (group)1
- *Measure Names*
- Age
- Blood Pressure
- BMI
- Diabetes Pedigree Function
- Glucose
- Insulin
- Outcome
- Pregnancies
- Skin Thickness
- *health care diabetes (Cou...*
- *Measure Values*

Pages

Columns

Rows

Filters

Age (bin)

Marks

Circle

Color Size Label

Detail Tooltip

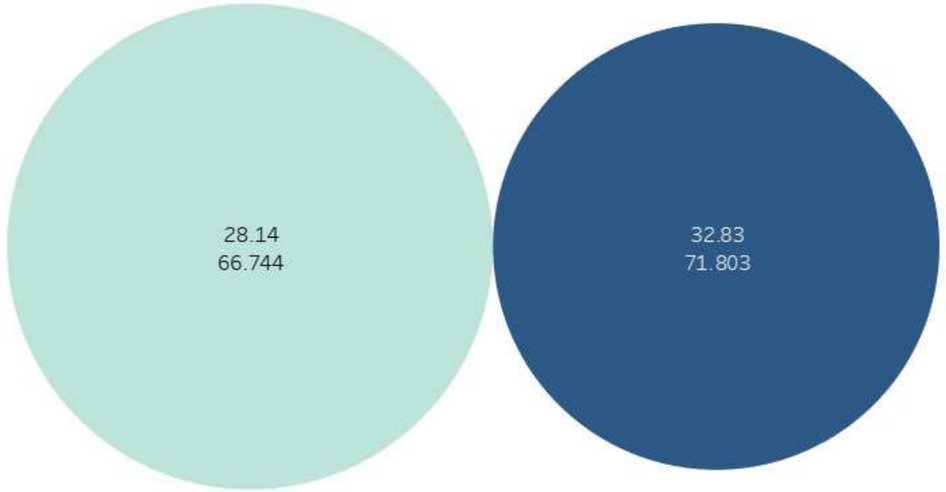
AVG(Blood Pr..

CNT(Age)

Age (bin)

AVG(Blood Pr..

Age(30-35) ,Avg Blood Pressure level Graph using Bubble Chart.



AVG(Blood Pressure)

66.744 71.803

Data Source Sheet 1 Sheet 2 Sheet 3 Sheet 4 Sheet 5 Sheet 6 **Sheet 7** Sheet 8 Sheet 9 Sheet 10 Sheet 11

2 marks 1 row by 1 column SUM of AVG(Blood Pressure): 138.547

Tableau Public - health

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Standard Show Me

Data Analytics

health care diabetes (he...

Search

Tables

- Age (bin)
- Outcome (bin)
- Outcome (group)1
- *Measure Names*
- Age
- Blood Pressure
- BMI
- Diabetes Pedigree Function
- Glucose
- Insulin
- Outcome
- Pregnancies
- Skin Thickness
- *health care diabetes (Cou...*
- *Measure Values*

Pages

Columns

Rows

Filters

Age (bin)

Marks

Circle

Color Size Label

Detail Tooltip

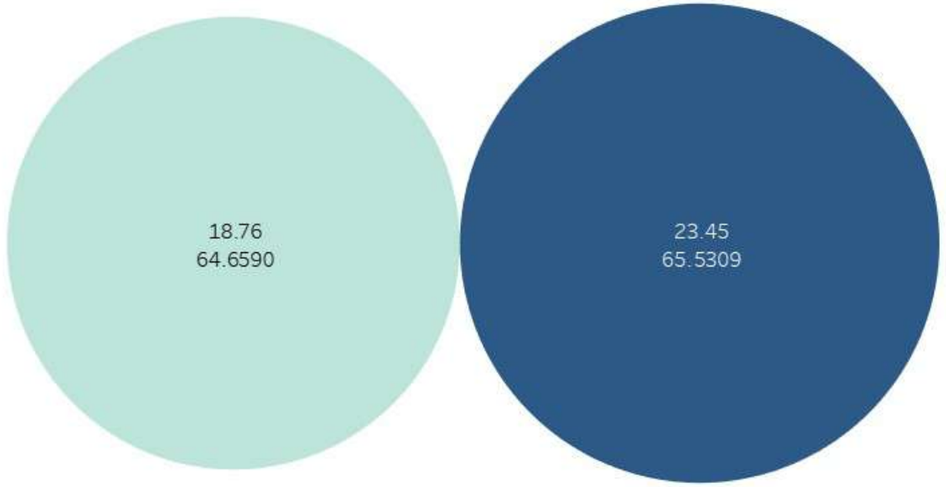
AVG(Blood Pr..

CNT(Age)

Age (bin)

AVG(Blood Pr..

Age(20-25) ,Avg Blood Pressure Level Graph using Bubble Chart.



AVG(Blood Pressure)

64.6590 65.5309

Data Source Sheet 1 Sheet 2 Sheet 3 Sheet 4 Sheet 5 Sheet 6 Sheet 7 Sheet 8 Sheet 9 Sheet 10 Sheet 11

2 marks 1 row by 1 column SUM of AVG(Blood Pressure): 130.1899

Tableau Public - health

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Standard Show Me

Data Analytics

health care diabetes (he...

Search

Tables

- Age (bin)
- Outcome (bin)
- Outcome (group)1
- *Measure Names*
- Age
- Blood Pressure
- BMI
- Diabetes Pedigree Function
- Glucose
- Insulin
- Outcome
- Pregnancies
- Skin Thickness
- *health care diabetes (Cou...*
- *Measure Values*

Pages

Columns

Rows

Filters

Age (bin)

Marks

Circle

Color Size Label

Detail Tooltip

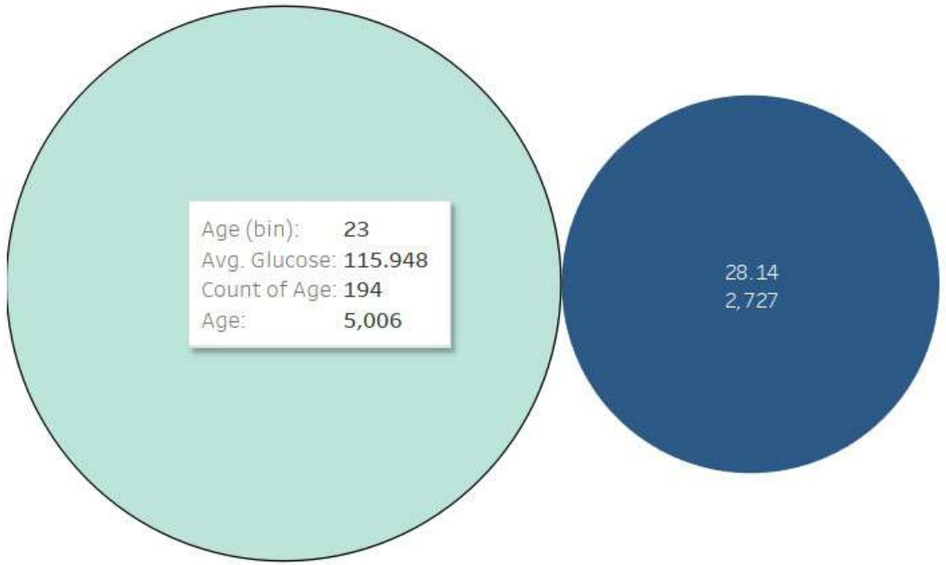
CNT(Age)

Age (bin)

SUM(Age)

AVG(Glucose)

Age(25-30) ,Avg Glucose Level Graph using Bubble Chart.



AVG(Glucose)

115.948 124.111

Data Source | Sheet 1 | Sheet 2 | Sheet 3 | Sheet 4 | Sheet 5 | Sheet 6 | Sheet 7 | Sheet 8 | **Sheet 9** | Sheet 10 | Sheet 11

2 marks 1 row by 1 column SUM(Age): 7.733

Tableau Public - health

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Standard Show Me

Data Analytics

health care diabetes (he...

Search

Tables

- Age (bin)
- Outcome (bin)
- Outcome (group)1
- *Measure Names*
- Age
- Blood Pressure
- BMI
- Diabetes Pedigree Function
- Glucose
- Insulin
- Outcome
- Pregnancies
- Skin Thickness
- *health care diabetes (Cou...*
- *Measure Values*

Pages

Columns

Rows

Filters

Age (bin)

Marks

Circle

Color Size Label

Detail Tooltip

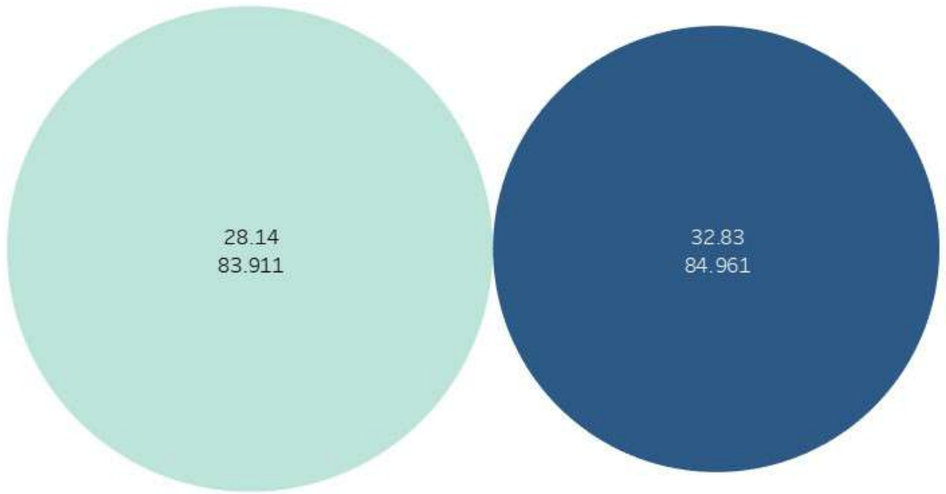
AVG(Insulin)

CNT(Age)

Age (bin)

AVG(Insulin)

Age(30-35) ,Avg insulin Level Graph using Bubble Chart.



AVG(Insulin)

83.911 84.961

Data Source | Sheet 1 | Sheet 2 | Sheet 3 | Sheet 4 | Sheet 5 | Sheet 6 | Sheet 7 | Sheet 8 | Sheet 9 | Sheet 10 | Sheet 11

2 marks 1 row by 1 column SUM of AVG(Insulin): 168.872

Tableau Public - health

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Show Me

Dashboard Layout

Default
Phone
Device Preview

Size
Automatic

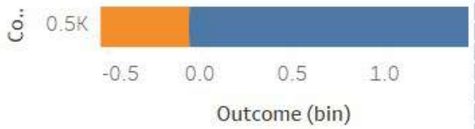
Sheets
Sheet 4
Sheet 5
Sheet 6
Sheet 7
Sheet 8
Sheet 9
Sheet 10

Objects
Ask Data
Data Story
Image
Blank
Workflow
Web Page
Navigation
Download
Add Filters
Einstein Discovery

Tiled Floating

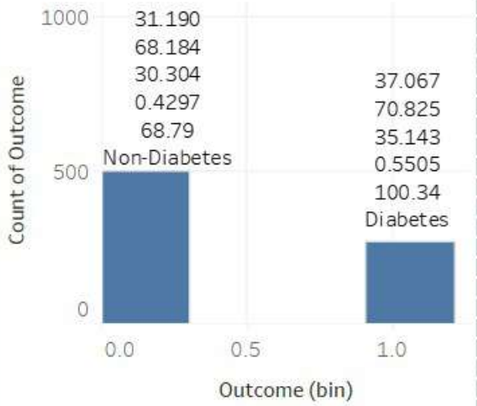
Show dashboard title

Pie chart to describe the diabetic or non-diabetic population

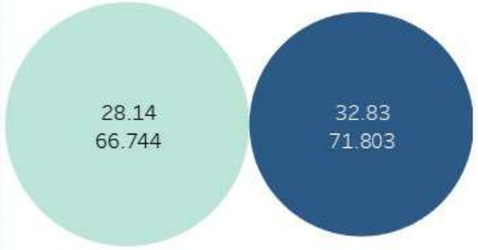


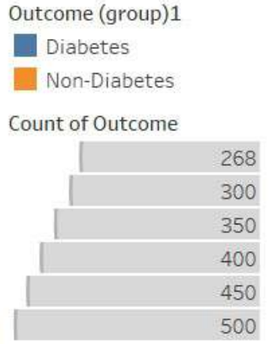
Age(20-25) ,Avg Glucose Level Graph using Bubble Chart.

Histogram or frequency charts to analyze the distribution of the data

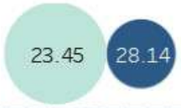


Age(30-35) ,Avg Blood Pressure level Graph using Bubble Chart.



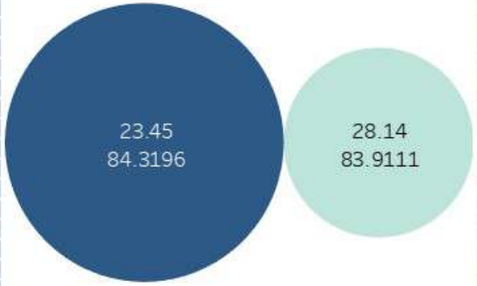


Age(25-30) ,Avg Glucose Level Graph using Bubble Chart.

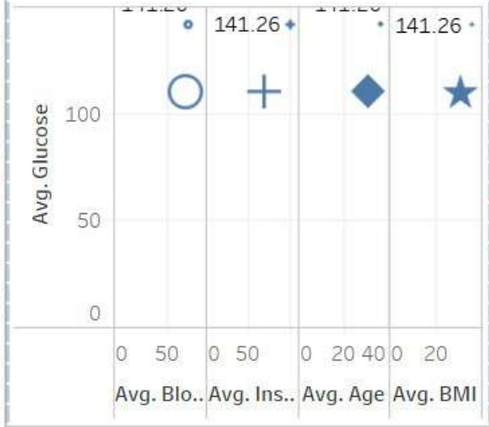


Sheet 3

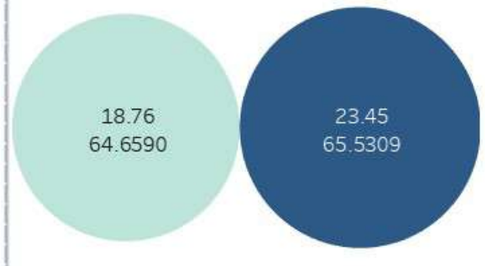
Age(25-30) ,Avg Insulin Level Graph using Bubble Chart.



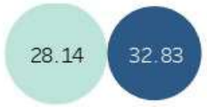
Scatter charts between relevant variables to analyze the relationships



Age(20-25) ,Avg Blood Pressure Level Graph using Bubble Chart.



Age(30-35) ,Avg insulin Level Graph using Bubble Chart.



Data Source | Sheet 1 | Sheet 2 | Sheet 3 | Sheet 4 | Sheet 5 | Sheet 6 | Sheet 7 | Sheet 8 | Sheet 9 | Sheet 10 | Dashboard 1

Tableau Public - health

File Data Worksheet Dashboard Story Analysis Map Format Server Window Help

Entire View | Show Me

Data | Analytics

health care diabetes (he...

Search

Tables

- Age (bin)
- Outcome (bin)
- Outcome (group)1
- *Measure Names*
- Age
- Blood Pressure
- BMI
- Diabetes Pedigree Function
- Glucose
- Insulin
- Outcome
- Pregnancies
- Skin Thickness
- *health care diabetes (Cou...*
- *Measure Values*

Pages

Filters

Marks

All

Multiple

Color | Size | Label | Detail | Tooltip | Shape

Outcome (gro..

Outcome (gro..

AVG(Blood Pre...

AVG(Insulin)

AVG(Age)

AVG(BMI)

Columns: AVG(Blood Pressure) | AVG(Insulin) | AVG(Age) | AVG(BMI)

Rows: AVG(Glucose)

Scatter charts between relevant variables to analyze the relationships

Outcome (group)1

- Diabetes
- Non-Diabetes

Data Source | Sheet 1 | Sheet 2 | Sheet 3 | **Sheet 4** | Sheet 5 | Sheet 6 | Sheet 7 | Sheet 8 | Sheet 9 | Sheet 10 | Dashboard 1

marks | 1 row by 4 columns | SUM of AVG(Age): 68.257