Predikcia Cukrovky pomocou algoritmov strojového učenia

Vypracovali Michal Knor a Martin Jankech

Tento dataset pochádza z Národného inštitútu pre diabetes a choroby tráviaceho traktu a obličiek (USA).
Cieľom je na základe diagnostických meraní predpovedať, či má pacient diabetes.
Rizikové faktory pre cukrovku sú:

- výskyt cukrovky v priamom príbuzenstve
- Vek. Zatiaľ čo pre diabetes 1. typu je charakteristický začiatok v detskom a mladom dospelom veku, výskyt diabetu 2. typu stúpa s vekom, najmä po 40. roku života
- výskyt zvýšených hodnôt krvného cukru v minulosti
- výskyt cukrovky v tehotenstve
- zvýšená telesná hmotnosť alebo tučnota
- zvýšené hodnoty krvného tlaku
- zvýšené hodnoty krvných tukov
- stres a/alebo depresia
- ochorenie srdca v prítomnosti alebo minulosti.

In [1]:
```
# import všetkých potrebných knižníc
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
from scipy import stats
from sklearn import naive_bayes
import warnings
warnings.filterwarnings('ignore')
```

1. načítanie dát + exploratívna analýza

- údaje boli oddelené ; preto bolo treba použiť vo funkcii read_csv parameter sep=";"

In [2]:
```
# nacitanie datasetu z csv súboru
data = pd.read_csv(r"C:\\Users\\janke\OneDrive\\Počítač\\škola 5 ročník\\1 semester\\Maschine learning\\projekt\diabetes.csv",sep=';')
# vypísanie hlavičky s prvými piatimi údajmi
data.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

Opis údajov z datasetu

- Stĺpec Pregnancies znamená koľkokrát bola pacientka tehotná
- Stĺpec Glukose sa vzťahuje plazmatickú koncentráciu glukózy 2 hodiny pri orálnom glukózovom tolerančnom teste. Podľa hodnôt odhadujeme že merna jednotka je mg/dL
- Stĺpec BloodPressure sa vzťahuje na Diastolický krvný tlak (mm Hg) pacientov.
- Stĺpec SkinThickness kože sa vzťahuje na hrúbku kožného záhybu tricepsu (mm)
- Stĺpec Insulin inzulínu sa týka použitia inzulínu u pacientov merná jednotka mIU/L
- Stĺpec BMI sa vzťahuje na index telesnej hmotnosti pacientov
- Stĺpec DiabestesPedigreeFunction označuje funkciu, ktorá hodnotí pravdepodobnosť diabetu na základe rodinnej anamnézy.
- Stĺpec age sa vzťahuje na vek pacientov
- Stĺpec Outcome znamená, že ak má pacient cukrovku (1) ak nemá (0) - predstavuje našu závislú premennú

In [3]:
```
data.shape
```

Out[3]:
```
(768, 9)
```

In [4]:
```
data.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]:
```
# základne štatistické údaje

data.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

Pozorovanie

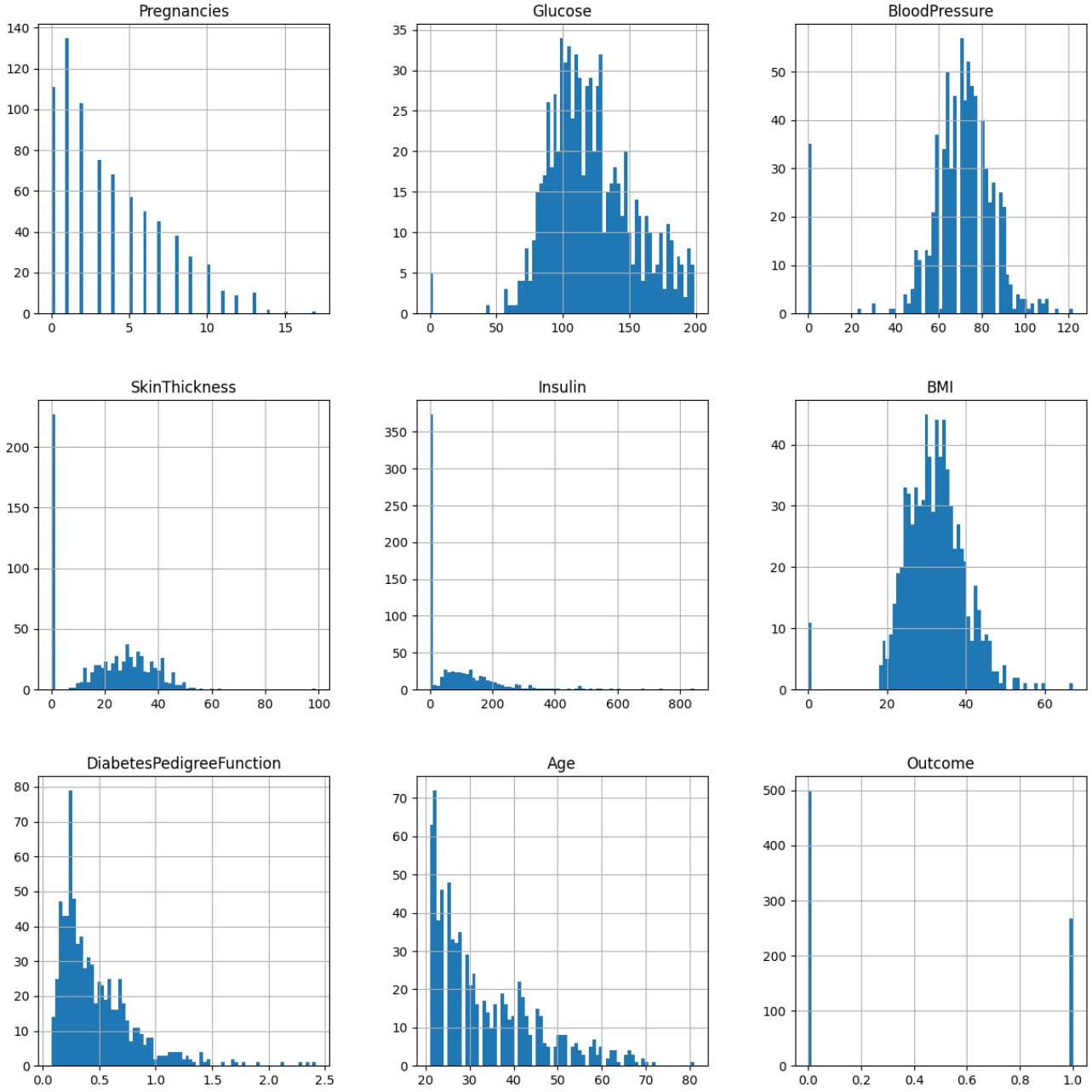
- vysoká smerodajná odchýlka pri insuline, skinthickness a bloodpressure

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [7]:
```
# histogramy pred úpravou dat
data.hist(bins=75,figsize=(15,15))
```

Out[7]:
```
array([[<AxesSubplot: title={'center': 'Pregnancies'}>,
        <AxesSubplot: title={'center': 'Glucose'}>,
        <AxesSubplot: title={'center': 'BloodPressure'}>],
       [<AxesSubplot: title={'center': 'SkinThickness'}>,
        <AxesSubplot: title={'center': 'Insulin'}>,
        <AxesSubplot: title={'center': 'BMI'}>],
       [<AxesSubplot: title={'center': 'DiabetesPedigreeFunction'}>,
        <AxesSubplot: title={'center': 'Age'}>,
        <AxesSubplot: title={'center': 'Outcome'}>]], dtype=object)
```



Pozorovanie

- Najviac pacientov je vo veku od 20 do 30 rokov
- Najviac pacientok ma 1 dieťa
- pomer pacientov, ktorý majú cukrovku k tým ktorým nebola diagnostikovaná je cca 1:2

data.hist(bins=75,figsize=(15,15))

Problémy

1. veľa hodnôt s 0 pri inzuline
2. veľa hodnôt s 0 pri Skinthickness
3. 0 pri glukoze, bloodpressure a bmi- predstavujú nereálne údaje

Mazanie nevhodných údajov

Rozhodli sme sa vymazať všetky hodnoty, ktoré mali pri insuline alebo skinthickness 0 - problémom je akurát že takto odstránime cca 300 hodnôt
Skúšali sme nahradiť 0 priemernou hodnotou, ale to taktiež nepomohlo, kedže tých 0 je celkom dosť- akurát to skreslilo rozdelenie
riešením by bolo aj odstránenie celých stĺpcov, ale tieto parametre považujeme za kľučové a preto sme sa ich rozhodli zachovať

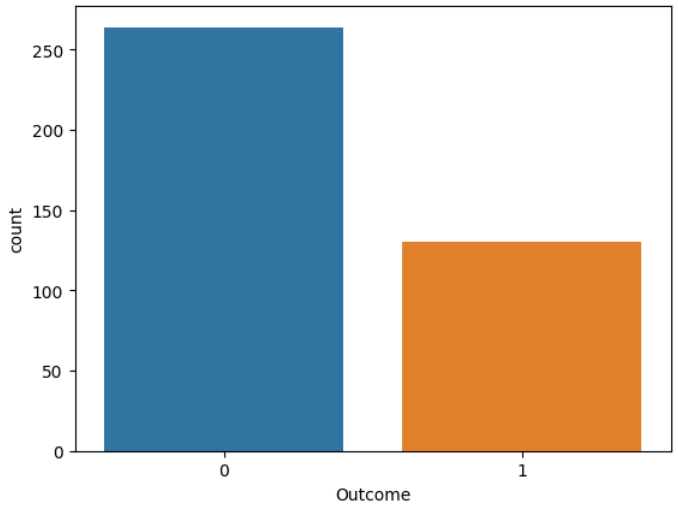
In [8]:
```
# odstranovanie nevhodnych hodnot
# odstranenie vsetkych hodnot ktore maju  v insuline alebo skinthickness - problem zmazanych vyse 300 hodnot
data.drop(data[data['SkinThickness'] == 0].index, inplace=True)
data.drop(data[data['Insulin'] == 0].index, inplace=True)
# nahradenie 0 hodnot medianom - moc nepomohlo
#data['Insulin']=data['Insulin'].replace(0,data['Insulin'].mean())
#data['SkinThickness']=data['SkinThickness'].replace(0,data['SkinThickness'].mean())
data
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
6	3	78	50	32	88	31.0	0.248	26	1
8	2	197	70	45	543	30.5	0.158	53	1
13	1	189	60	23	846	30.1	0.398	59	1
...	...	...	...	...	...	...	...	...	...
753	0	181	88	44	510	43.3	0.222	26	1
755	1	128	88	39	110	36.5	1.057	37	1
760	2	88	58	26	16	28.4	0.766	22	0
763	10	101	76	48	180	32.9	0.171	63	0
765	5	121	72	23	112	26.2	0.245	30	0

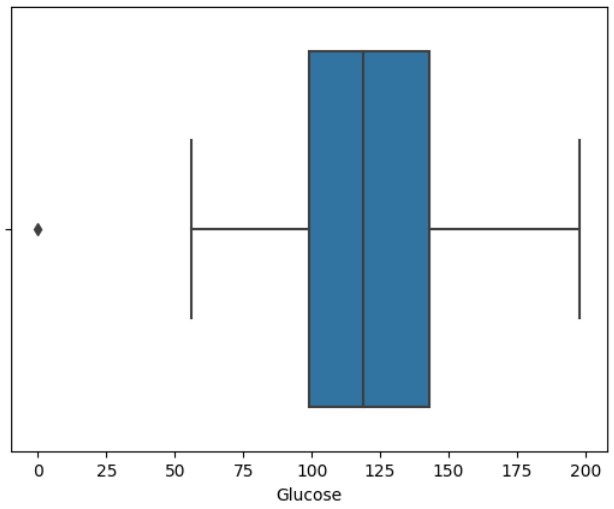
394 rows × 9 columns

In [9]:
```
sns.countplot(x=data['Outcome'])
plt.show()
```



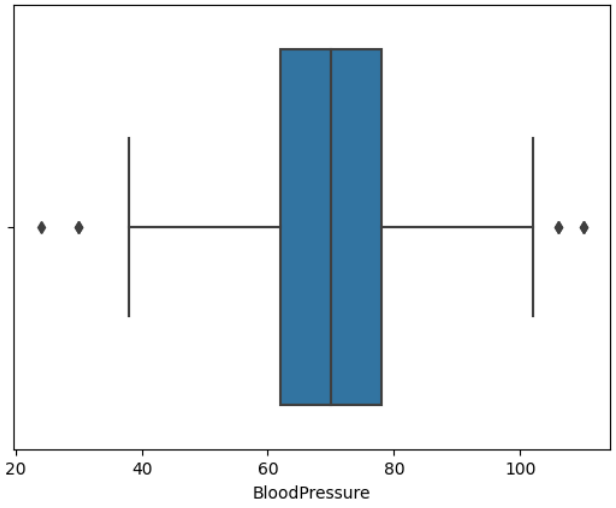
Odstránenie odľahlých pozorovaní

In [10]:
```
sns.boxplot(x=data['Glucose'])
plt.show()
```

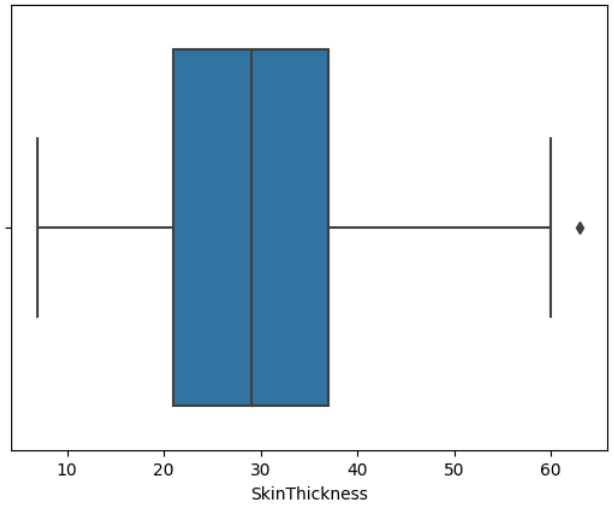


In [11]:
```
data.drop(data[data['Glucose'] == 0].index, inplace=True)
```

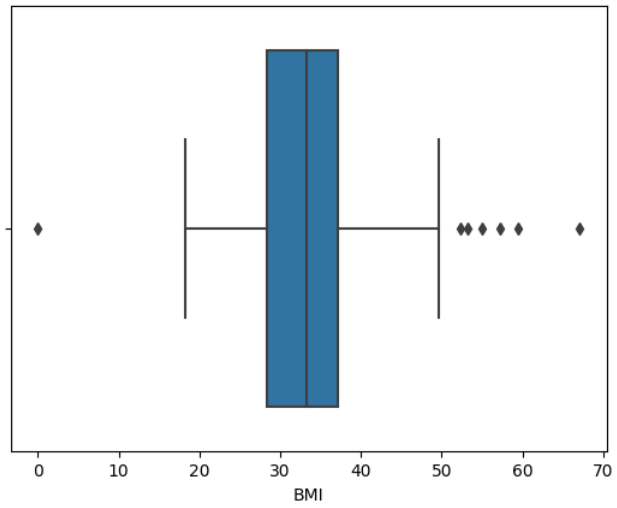
In [12]:
```
sns.boxplot(x=data['BloodPressure'])
plt.show()
data.drop(data[data['BloodPressure'] == 0].index, inplace=True)
```



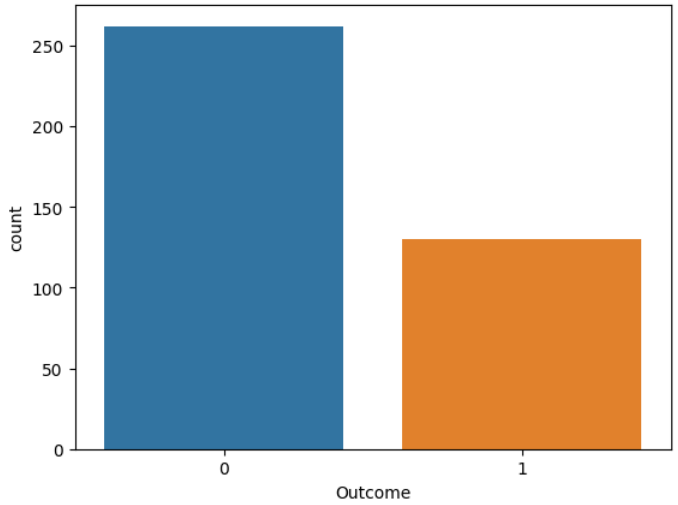
In [13]:
```
sns.boxplot(x=data['SkinThickness'])
plt.show()
data.drop(data[data['SkinThickness'] > 80].index, inplace=True)
```



In [14]:
```
sns.boxplot(x=data['BMI'])
plt.show()
data.drop(data[data['BMI'] == 0].index, inplace=True)
```

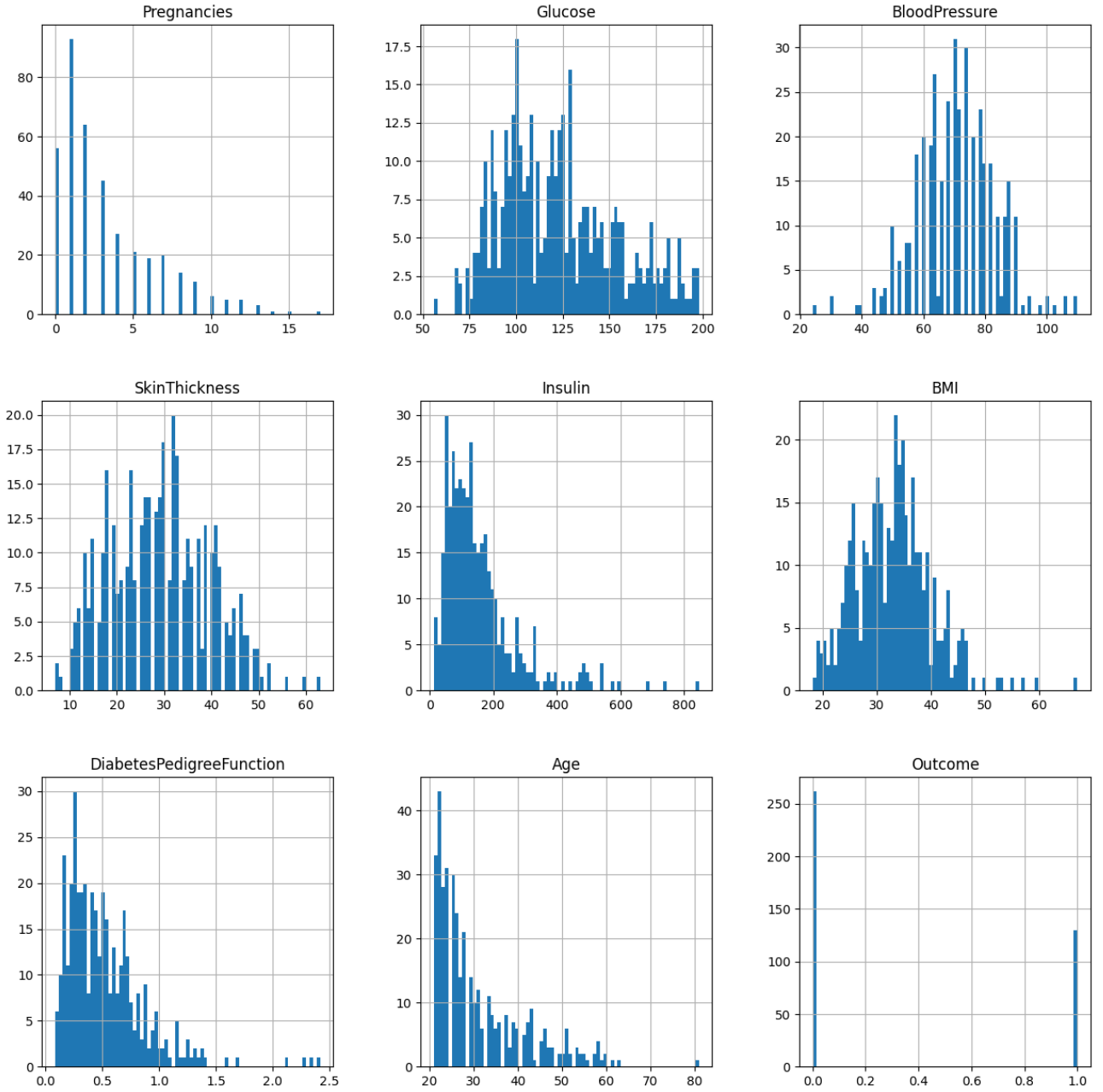


In [15]:
```
sns.countplot(x=data['Outcome'])
plt.show()
```



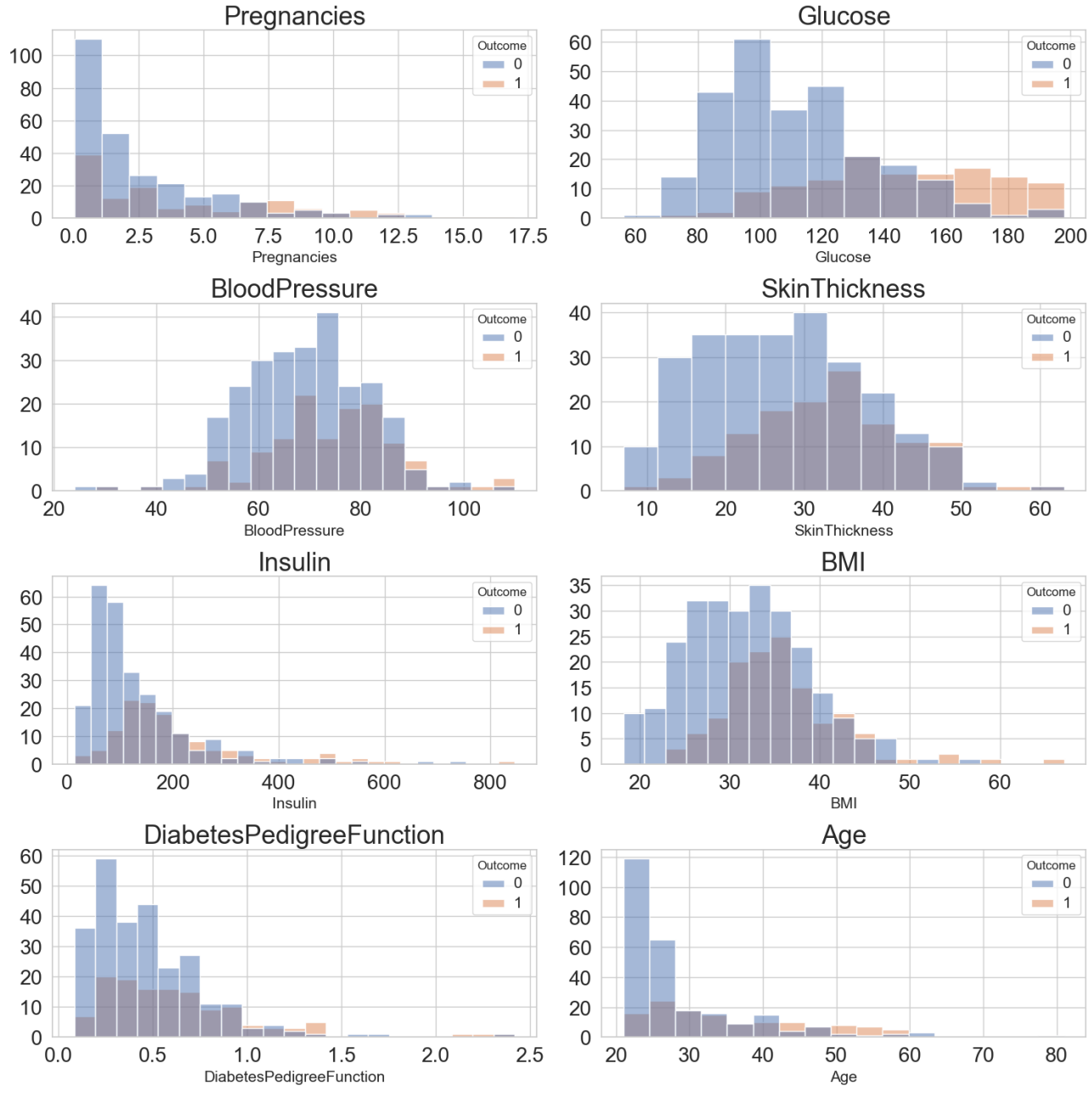
In [16]:
```
# histogramy po uprave dat
data.hist(bins=75,figsize=(15,15))
```

Out[16]:
```
array([[<AxesSubplot: title={'center': 'Pregnancies'}>,
        <AxesSubplot: title={'center': 'Glucose'}>,
        <AxesSubplot: title={'center': 'BloodPressure'}>],
       [<AxesSubplot: title={'center': 'SkinThickness'}>,
        <AxesSubplot: title={'center': 'Insulin'}>,
        <AxesSubplot: title={'center': 'BMI'}>],
       [<AxesSubplot: title={'center': 'DiabetesPedigreeFunction'}>,
        <AxesSubplot: title={'center': 'Age'}>,
        <AxesSubplot: title={'center': 'Outcome'}>]], dtype=object)
```



histogramy s outputmi

In [17]:
```
plt.figure(figsize=(15, 15))
sns.set(style='whitegrid')
sns.diverging_palette(1000 ,500, l=65, center="dark", as_cmap=True)
plotnumber = 1

for feature in data.columns[:8]:
    ax = plt.subplot(4, 2, plotnumber)
    ax = sns.histplot(x = feature, hue = 'Outcome', data = data)
    plt.setp(ax.get_legend().get_texts(), fontsize='15')
    plt.xlabel(feature, size = 15)
    plt.title(feature, size = 25)
    plt.xticks(size = 20)
    ax.set_ylabel('')
    plt.yticks(size = 20)
    plotnumber += 1
    plt.tight_layout()
    ;
```



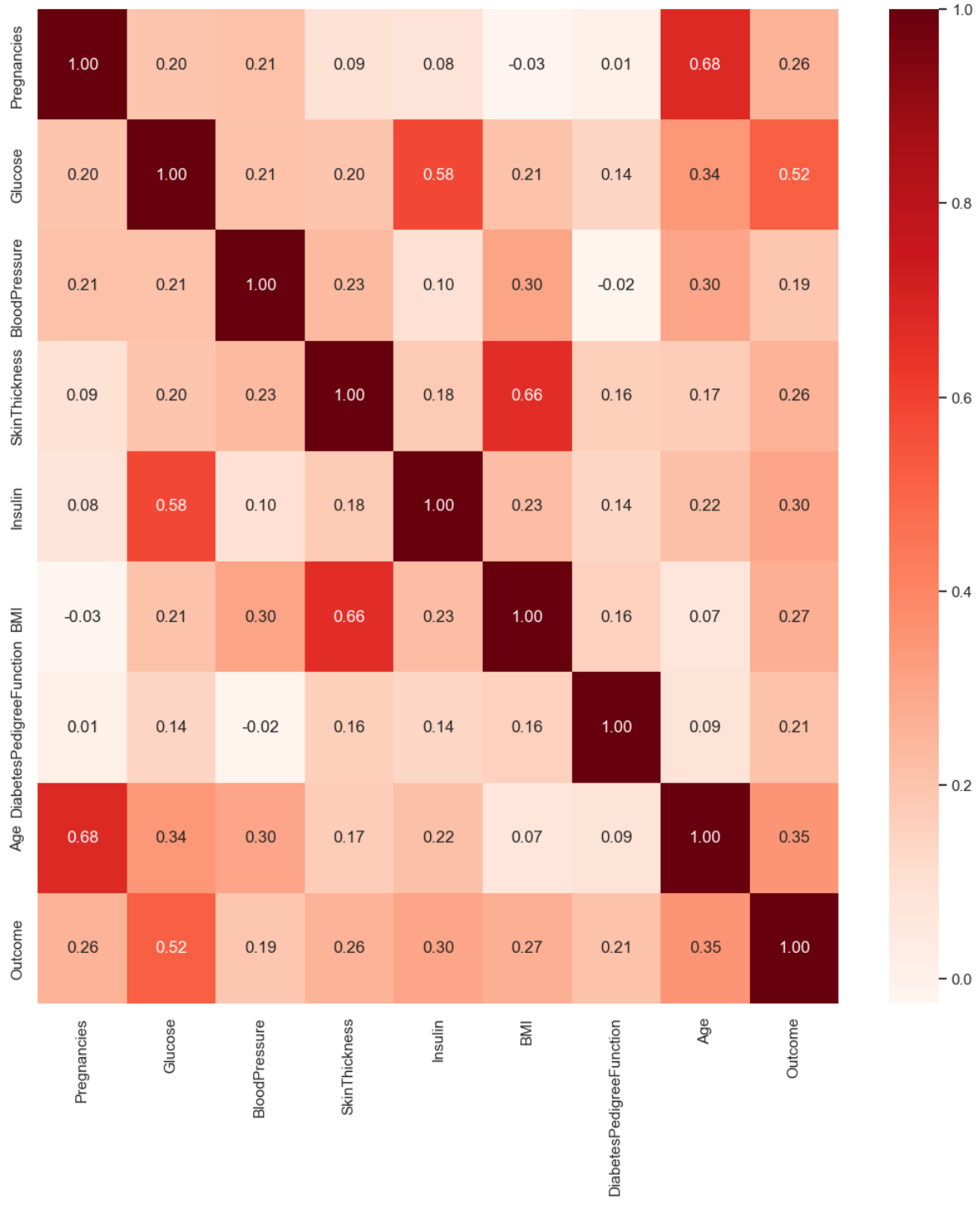
Pozorovanie

vačší počet ľudi s cukrovkov(1) oproti ľudom bez(0) bol zistený

1. u žien s viac ako 7 detmi
2. osobách s hladinou glukózy vačšiou ako 160
3. bmi nad 40
4. vek nad 40
5. DiabestesPedigreeFunction nad 1

In [18]:
```
# vyber nezavislych atributov s ktorými budeme dalej pracovať pri aplikovani ML algoritmov - pre zlepsenie accuracy sme skušali viacero variant
feature_names = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI", "DiabetesPedigreeFunction", "Age"]
```

```
#feature_names = ["Glucose", "BloodPressure",  "BMI", "DiabetesPedigreeFunction", "Age"]
#eature_names = ["Pregnancies", "Glucose", "BloodPressure",  "BMI", "DiabetesPedigreeFunction", "Age"]

plt.figure(figsize=(13, 13))
cor = data.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds, fmt='.2f')
plt.show()
```



In [19]:
```
x = pd.DataFrame(data, columns=feature_names)
y = data.Outcome.values.reshape(-1, 1)
print(x)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
6              3       78             50             32       88  31.0
8              2      197             70             45      543  30.5
13             1      189             60             23      846  30.1
..           ...      ...            ...            ...      ...   ...
753            0      181             88             44      510  43.3
755            1      128             88             39      110  36.5
760            2       88             58             26       16  28.4
763           10      101             76             48      180  32.9
765            5      121             72             23      112  26.2

     DiabetesPedigreeFunction  Age
3                       0.167   21
4                       2.288   33
6                       0.248   26
8                       0.158   53
13                      0.398   59
..                        ...  ...
753                     0.222   26
755                     1.057   37
760                     0.766   22
763                     0.171   63
765                     0.245   30

[392 rows x 8 columns]
```

Pozorovanie

- stredne silná korelácia outputu a glukózy
- pri ostatných slabá korelácia
- pri nezávislých premenných silná korelácia počtu tehotenstviev a veku ako aj bmi a skinthickness

3. Rozdelenie datasetu na trénovaciu a testovaciu časť

In [20]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,random_state=125)
```

4. Aplikácia Ml algoritmov a porovnanie ich výkonnosti

Popis algoritmov

Rozhodovaci strom

- Výhody - Používa model bielej skrinky. Ak je daná situácia pozorovateľná v modeli, vysvetlenie podmienky sa dá ľahko vysvetliť boolovskou logikou. Naopak, v modeli čiernej skrinky (napr. v umelej neurónovej sieti) môže byť interpretácia výsledkov zložitejšia
- Nevýhody - Rozhodovacie stromy môžu byť nestabilné, pretože malé odchýlky v údajoch môžu viesť k vygenerovaniu úplne iného stromu.

parameter gini - Nečistota gini meria frekvenciu, pri ktorej bude akýkoľvek prvok súboru údajov nesprávne označený, keď je náhodne označený.
Minimálna hodnota Gini indexu je 0. Stáva sa to, keď je uzol čistý, to znamená, že všetky obsiahnuté prvky v uzle sú z jednej jedinečnej triedy.
max_depht
Maximálna hĺbka stromu. Ak nie je, potom sa už rozširujú, kým nie sú všetky listy čisté alebo kým všetky listy neobsahujú menej ako min_samples_split vzoriek.

Support Vector Machine

- Základom metódy SVM je lineárny klasifikátor do dvoch tried. Cieľom úlohy je nájsť nadrovinu, ktorá priestor príznakov optimálne rozdeľuje tak, že trénovacie dáta patriace odlišným triedam ležia v opačných polopriestoroch. Optimálna nadrovina je taká, že hodnota minima vzdialeností bodov od roviny je čo najväčšia.

Gaussian Radial Basis Function (RBF)-Je to jedna z najpreferovanejších a najpoužívanejších funkcií jadra v svm.
oproti linearnemu nam zlepsilo accuracy o 5percent

logicticka regresia
nižsia accuracy pretože niektoré nezávislé premenné majú vyššiu vzajomnú koreláciu

random forest
rozhodovacie lesy je súborová metóda učenia pre klasifikáciu, regresiu a iné úlohy, ktorá funguje tak, že v čase učenia vytvára množstvo rozhodovacích stromov.
Pre klasifikačné úlohy je výstupom náhodného lesa trieda, ktorú vyberie väčšina stromov

KNeighborsClassifier
vyšla nám lepšia accuracy pri párnej hodnote k=6 aj keď správne by bolo použiť nepárnu k hodnotu. napr. defaultnú 5

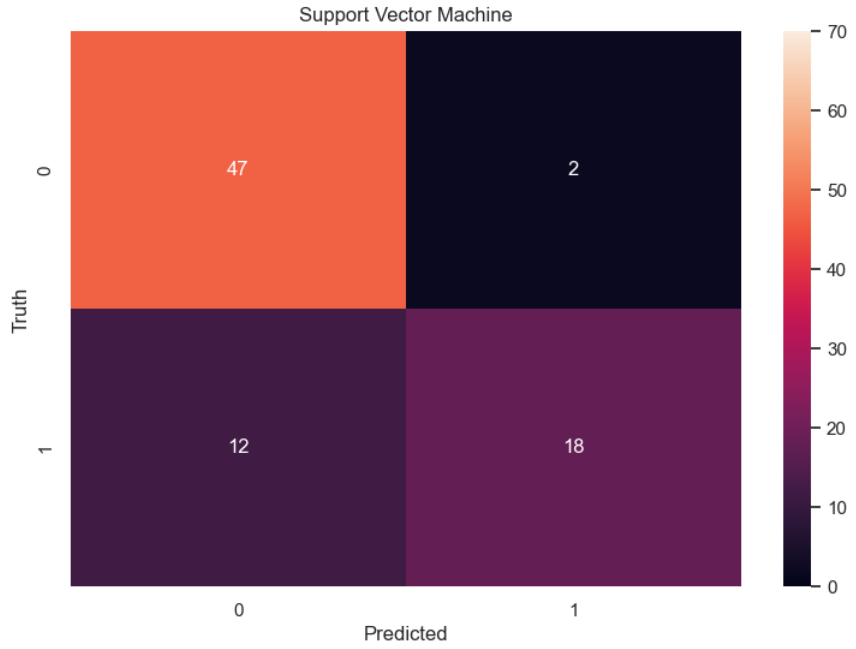
In [21]:
```
from sklearn import svm
```

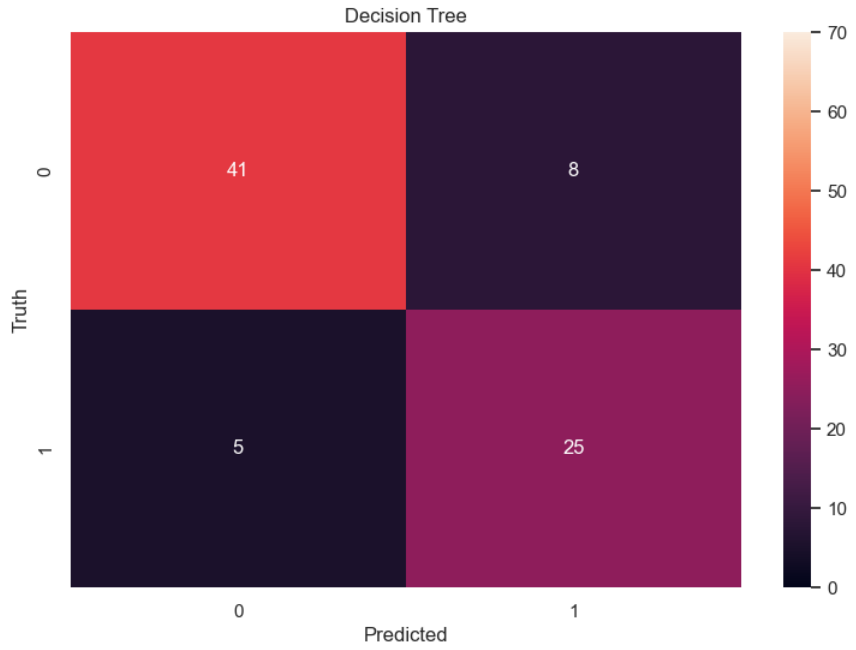
In [22]:
```
algorithms = ["Support Vector Machine",
              "Decision Tree",
              "GradientBoostingClassifier",
              "log res",
             "KNeighborsClassifier",
             "RandomForestClassifier"]
```

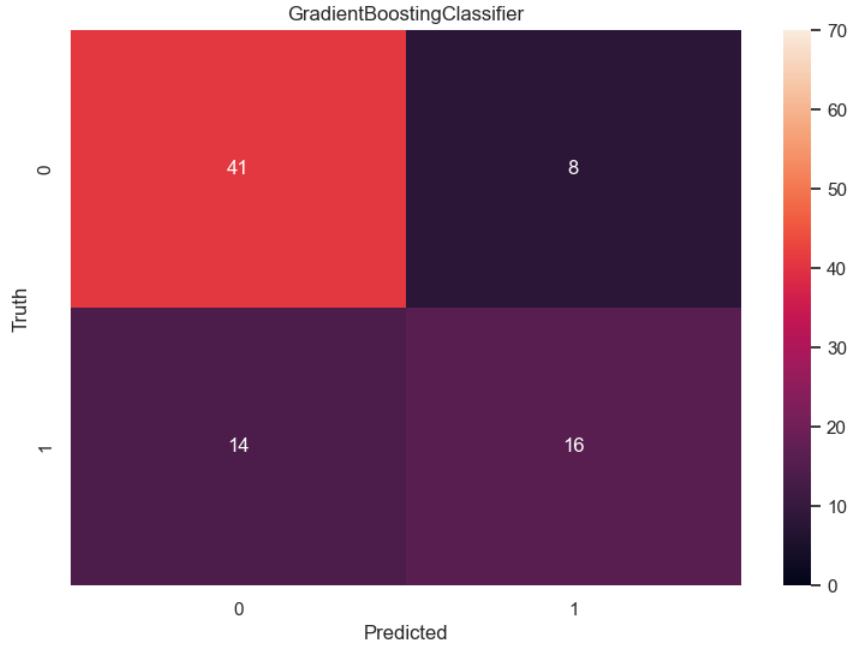
In [23]:
```
from sklearn.linear_model import LogisticRegression
clfs = [svm.SVC(kernel="rbf"),
        DecisionTreeClassifier(criterion="gini", max_depth=3),
        GradientBoostingClassifier(n_estimators=150, max_depth=50, learning_rate=0.3, random_state=0),
        LogisticRegression(C=1.0, max_iter=100, multi_class="ovr", penalty="l1", solver="saga"),
       KNeighborsClassifier(6),
       RandomForestClassifier(max_depth=50, random_state=125)]
```

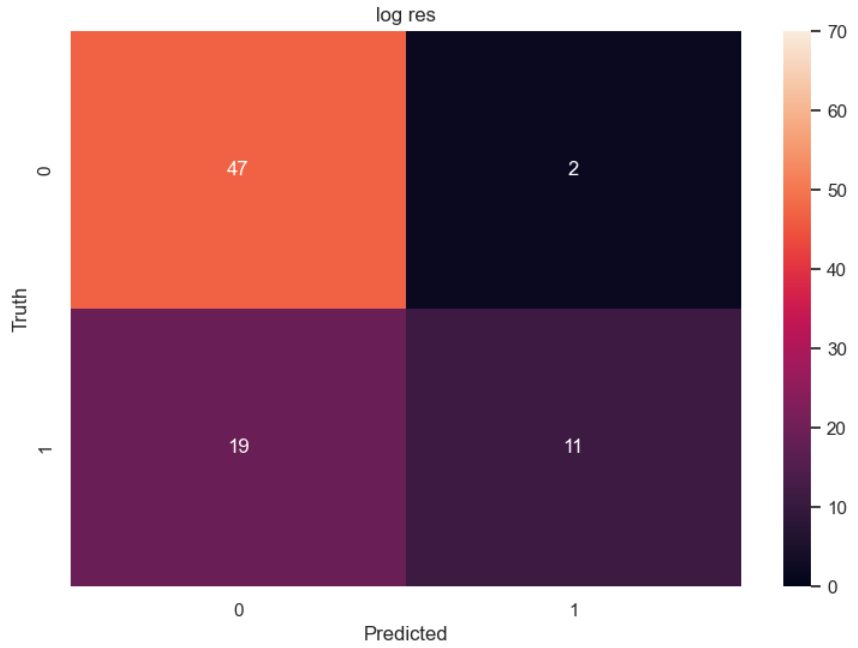
```
accuracies = []
clfs_result = {}
```

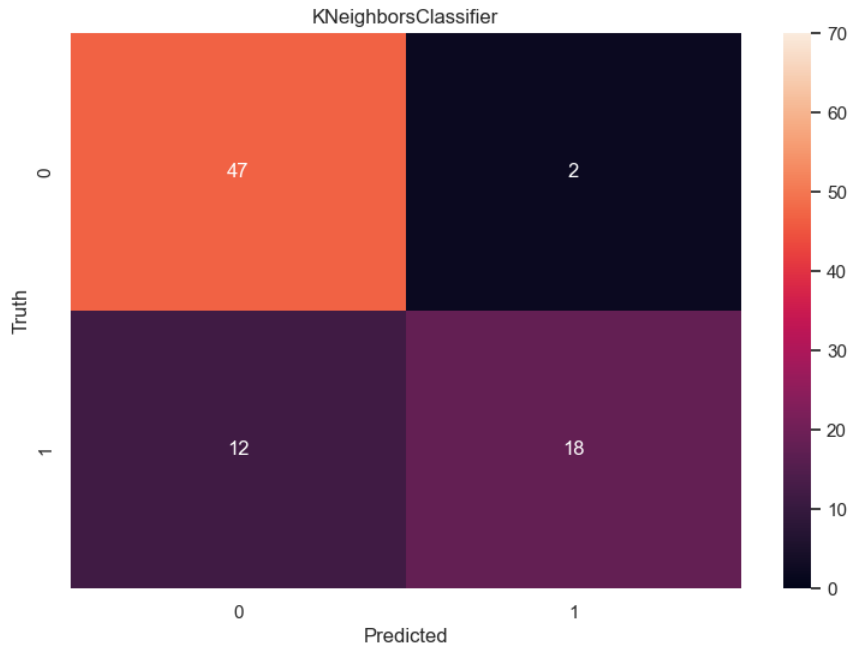
In [24]:
```
from sklearn.metrics import confusion_matrix
for i in range(len(clfs)):
    clf = clfs[i]
    algorithm = algorithms[i]

    clf = clf.fit(x_train, y_train)
    y_pred = clf.predict(x_test)
    acc = metrics.accuracy_score(y_test, y_pred)
    accuracies.append(acc)
    clfs_result[algorithm] = clf

    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(9, 6))
    sns.heatmap(cm, vmin=0, vmax=70, annot=True)
    plt.title(algorithm)
    plt.xlabel("Predicted")
    plt.ylabel("Truth")
    plt.show()
```

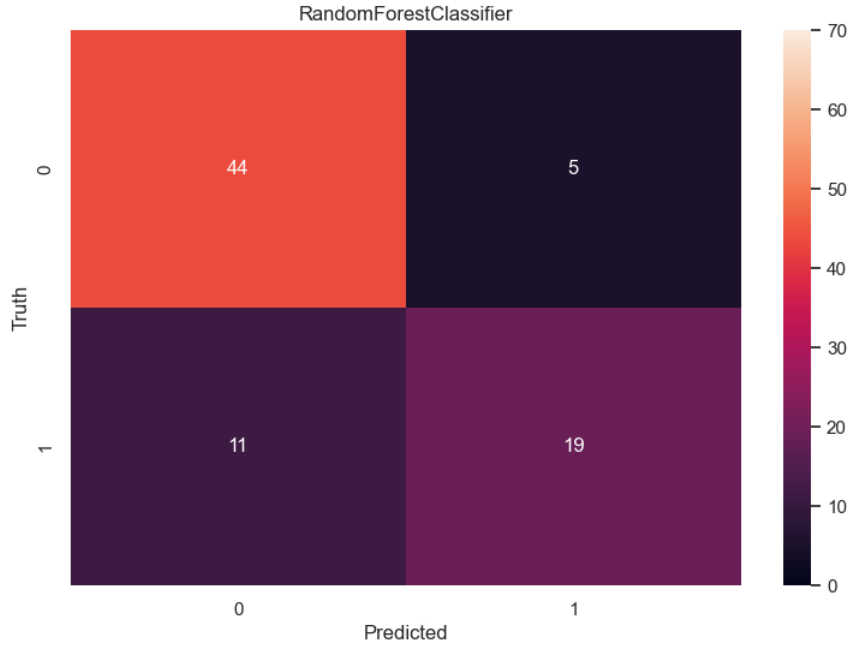





GradientBoostingClassifier
41
8
14
16
Truth
Predicted
0
1
70
60
50
40
30
20
10
0


log res
47
2
19
11
Truth
Predicted
0
1
70
60
50
40
30
20
10
0


KNeighborsClassifier
47
2
12
18
Truth
Predicted
0
1
70
60
50
40
30
20
10
0

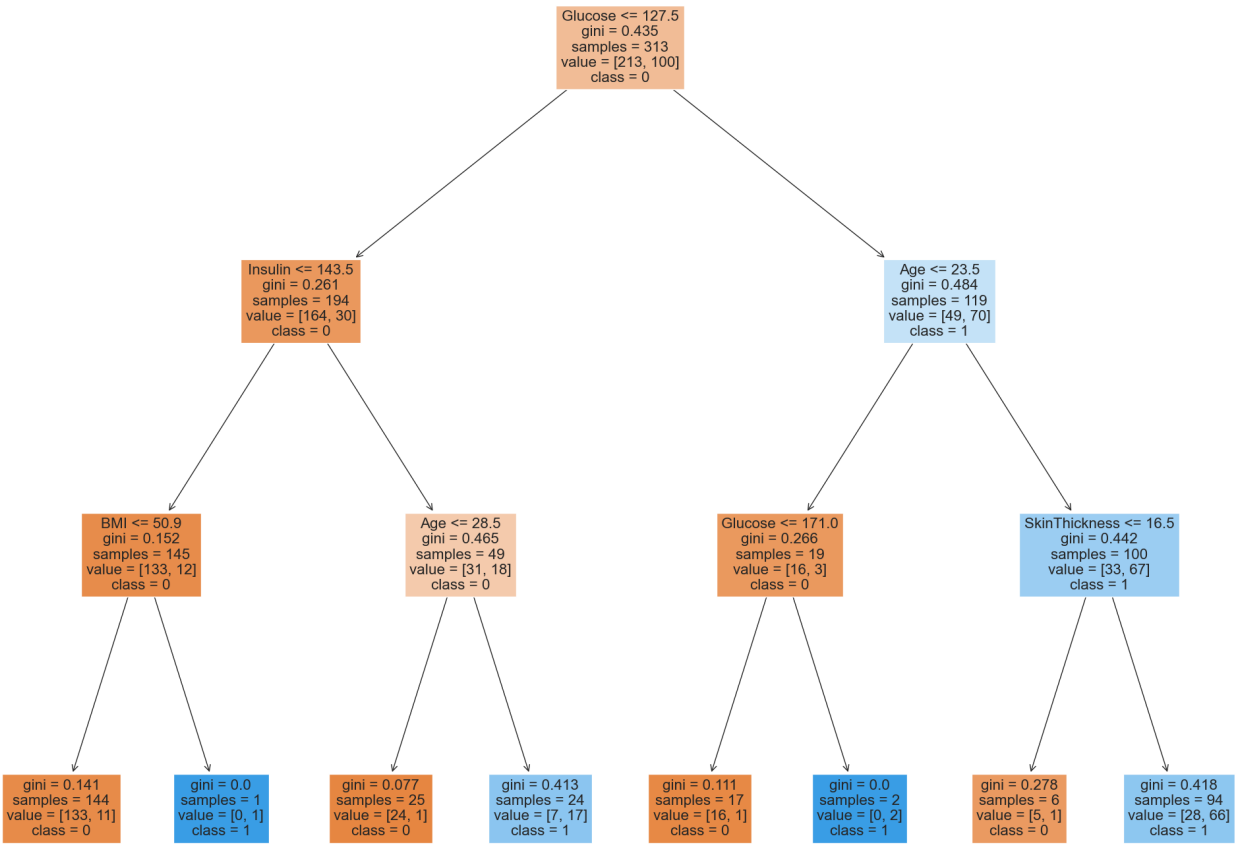


In [25]:
```
accuracies_df = pd.DataFrame(accuracies, index=algorithms)
accuracies_df.columns = ["Accuracy"]
print(accuracies_df.sort_values(by="Accuracy", ascending=False))
```

```
                           Accuracy
Decision Tree              0.835443
Support Vector Machine     0.822785
KNeighborsClassifier       0.822785
RandomForestClassifier     0.797468
log res                    0.734177
GradientBoostingClassifier 0.721519
```

In [26]:
```
from sklearn import tree
text_representation = tree.export_text(clfs_result["Decision Tree"])
print(text_representation)

target_names = ['0', '1']
fig = plt.figure(figsize=(25, 20))
plot = tree.plot_tree(clfs_result["Decision Tree"], feature_names=feature_names, class_names=target_names, filled=True)
plt.show()
```

```
|--- feature_1 <= 127.50
|   |--- feature_4 <= 143.50
|   |   |--- feature_5 <= 50.90
|   |   |   |--- class: 0
|   |   |--- feature_5 >  50.90
|   |   |   |--- class: 1
|   |--- feature_4 >  143.50
|   |   |--- feature_7 <= 28.50
|   |   |   |--- class: 0
|   |   |--- feature_7 >  28.50
|   |   |   |--- class: 1
|--- feature_1 >  127.50
|   |--- feature_7 <= 23.50
|   |   |--- feature_1 <= 171.00
|   |   |   |--- class: 0
|   |   |--- feature_1 >  171.00
|   |   |   |--- class: 1
|   |--- feature_7 >  23.50
|   |   |--- feature_3 <= 16.50
|   |   |   |--- class: 0
|   |   |--- feature_3 >  16.50
|   |   |   |--- class: 1
```



Predikovanie novej hodnoty všetkými algoritmami

In [27]:
```
#([Pregnacies,Glucoce,BloodPressure,SkinThickness,Insulin,BMI,DiabetesPedigreeFunction,Age,Outcome])
new_data_point=np.array([10,150,70,38,300,38,0.78,54]).reshape(1,-1)
#new_data_point1=np.array([2,40,30,20,30,11,0.13,20]).reshape(1,-1)
#new_data_point2=np.array([5,80,40,30,150,23,0.58,33]).reshape(1,-1)
new_data_point_pred=[]

for i in range(len(clfs)):
    clf = clfs[i]
    algorithm = algorithms[i]
    new_data_point_pred.append(clf.predict(new_data_point))

predicted_df = pd.DataFrame(new_data_point_pred, index=algorithms)
predicted_df.columns = ["predicted"]
print(predicted_df.sort_values(by="predicted", ascending=False))
```

```
                           predicted
Support Vector Machine             1
Decision Tree                      1
GradientBoostingClassifier         1
log res                            1
KNeighborsClassifier               1
RandomForestClassifier             1
```

In [ ]: