```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score,
GridSearchCV
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

diab_dataset = pd.read_csv("/content/diabetes.csv")
diab_dataset.head()
```


```
{"summary":"{\n  \"name\": \"diab_dataset\",\n  \"rows\": 768,\n
\"fields\": [\n    {\n      \"column\": \"Pregnancies\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 0,\n        \"max\": 17,\n
\"num_unique_values\": 17,\n        \"samples\": [\n          6,\n
1,\n          3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Glucose\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 31,\n        \"min\": 0,\n        \"max\": 199,\n
\"num_unique_values\": 136,\n        \"samples\": [\n          151,\n
101,\n          112\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BloodPressure\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 19,\n        \"min\": 0,\n
\"max\": 122,\n        \"num_unique_values\": 47,\n
\"samples\": [\n          86,\n          46,\n          85\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SkinThickness\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 15,\n        \"min\": 0,\n
\"max\": 99,\n        \"num_unique_values\": 51,\n        \"samples\":
[\n          7,\n          12,\n          48\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Insulin\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 115,\n
\"min\": 0,\n        \"max\": 846,\n        \"num_unique_values\":
186,\n        \"samples\": [\n          52,\n          41,\n
183\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BMI\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 7.884160320375446,\n        \"min\": 0.0,\n        \"max\":
67.1,\n        \"num_unique_values\": 248,\n        \"samples\": [\n
19.9,\n          31.0,\n          38.1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DiabetesPedigreeFunction\",\n
```

\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.3313285950127749,\n   \"min\": 0.078,\n   \"max\": 2.42,\n
\"num_unique_values\": 517,\n   \"samples\": [\n   1.731,\
n   0.426,\n   0.138\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 11,\n   \"min\": 21,\n
\"max\": 81,\n   \"num_unique_values\": 52,\n   \"samples\":
[\n   60,\n   47,\n   72\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Outcome\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 0,\n
\"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n
\"samples\": [\n   0,\n   1\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"diab_dataset"}

```
diab_dataset.shape
```

```
(768, 9)
```

```
diab_dataset.describe()
```

{"summary":"{\n \"name\": \"diab_dataset\",\n \"rows\": 8,\n
\"fields\": [\n   {\n   \"column\": \"Pregnancies\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
269.85223453356366,\n   \"min\": 0.0,\n   \"max\": 768.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
3.8450520833333335,\n   3.0,\n   768.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Glucose\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
243.73802348295857,\n   \"min\": 0.0,\n   \"max\": 768.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
120.89453125,\n   117.0,\n   768.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"BloodPressure\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
252.8525053581062,\n   \"min\": 0.0,\n   \"max\": 768.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
69.10546875,\n   72.0,\n   768.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"SkinThickness\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
263.7684730531098,\n   \"min\": 0.0,\n   \"max\": 768.0,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   768.0,\n
20.536458333333332,\n   32.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Insulin\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":

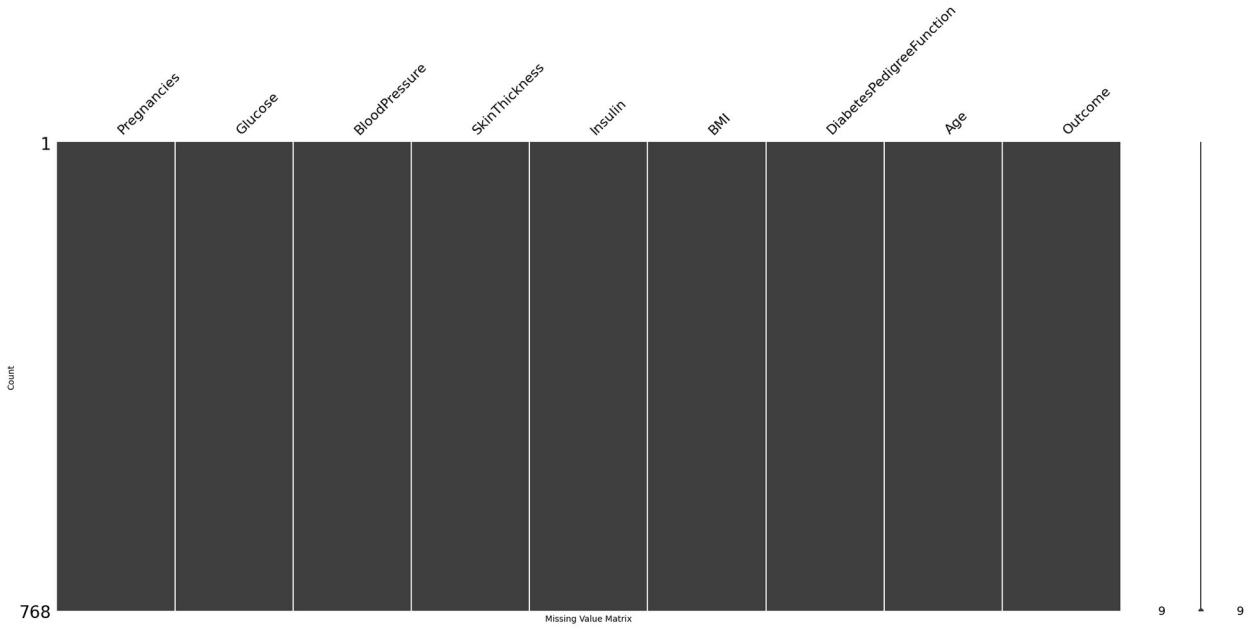
```
350.26059167945886,\n        \"min\": 0.0,\n        \"max\": 846.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          768.0,\n
79.79947916666667,\n          127.25\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"BMI\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 262.05117817552093,\n
\"min\": 0.0,\n        \"max\": 768.0,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          31.992578124999998,\n
32.0,\n          768.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"DiabetesPedigreeFunction\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 271.3005221658502,\n
\"min\": 0.078,\n        \"max\": 768.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
0.47187630208333325,\n          0.3725,\n          768.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 260.1941178528413,\n
\"min\": 11.760231540678685,\n        \"max\": 768.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
33.240885416666664,\n          29.0,\n          768.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Outcome\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
271.3865920388932,\n        \"min\": 0.0,\n        \"max\": 768.0,\n
\"num_unique_values\": 5,\n        \"samples\": [\n
0.3489583333333333,\n          1.0,\n          0.47695137724279896\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


```
# checking for null values
diab_dataset.isnull().sum()
```

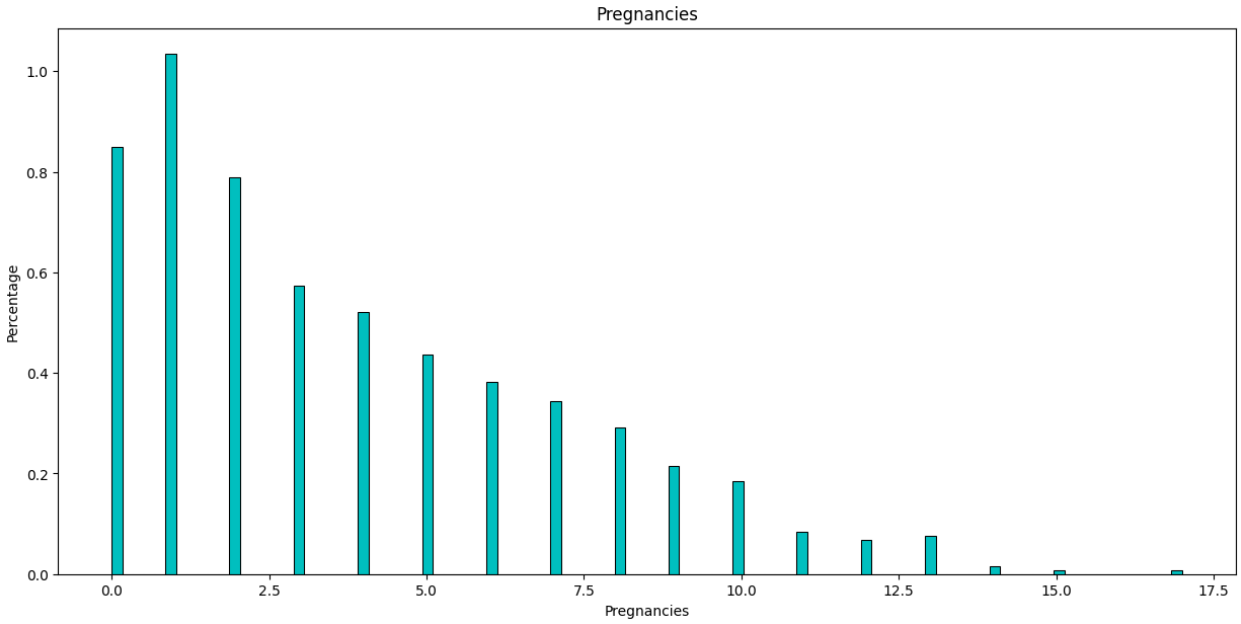
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
# Visualising Missing data
import missingno as msno
msno.matrix(diab_dataset)
plt.xlabel("Missing Value Matrix")
plt.ylabel("Count")
```

```
Text(0, 0.5, 'Count')
```

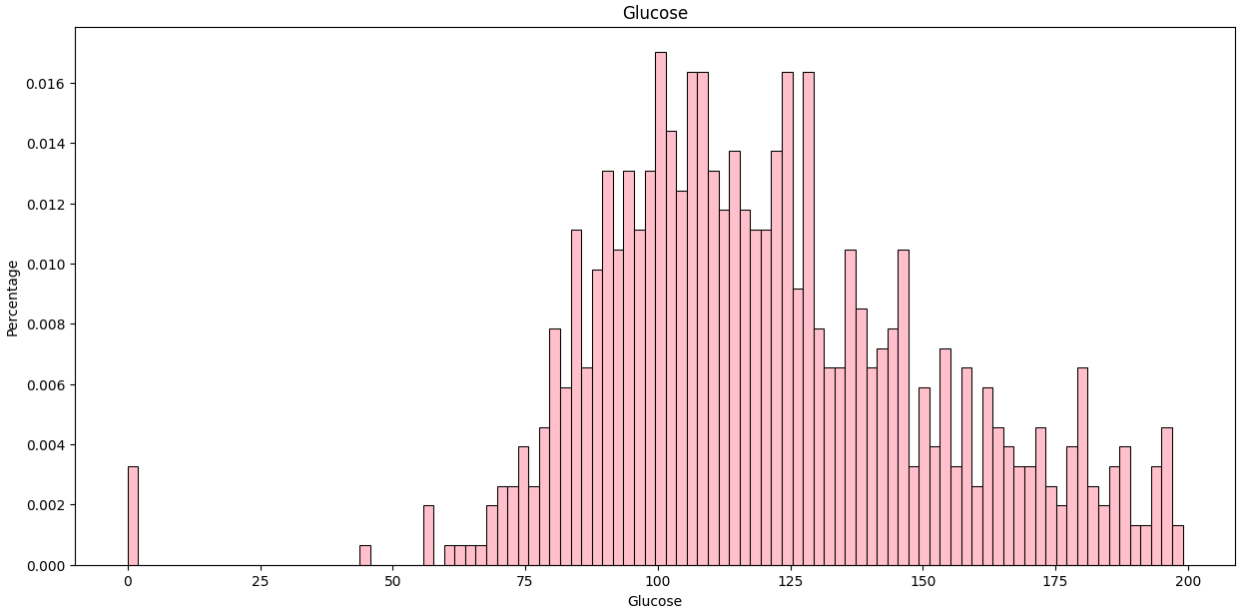



```
# Pregnancies
plt.figure(figsize=(15,7))
sns.histplot(diab_dataset["Pregnancies"], facecolor='c', bins=100,
stat="density");
plt.ylabel("Percentage")
plt.title("Pregnancies")

Text(0.5, 1.0, 'Pregnancies')
```



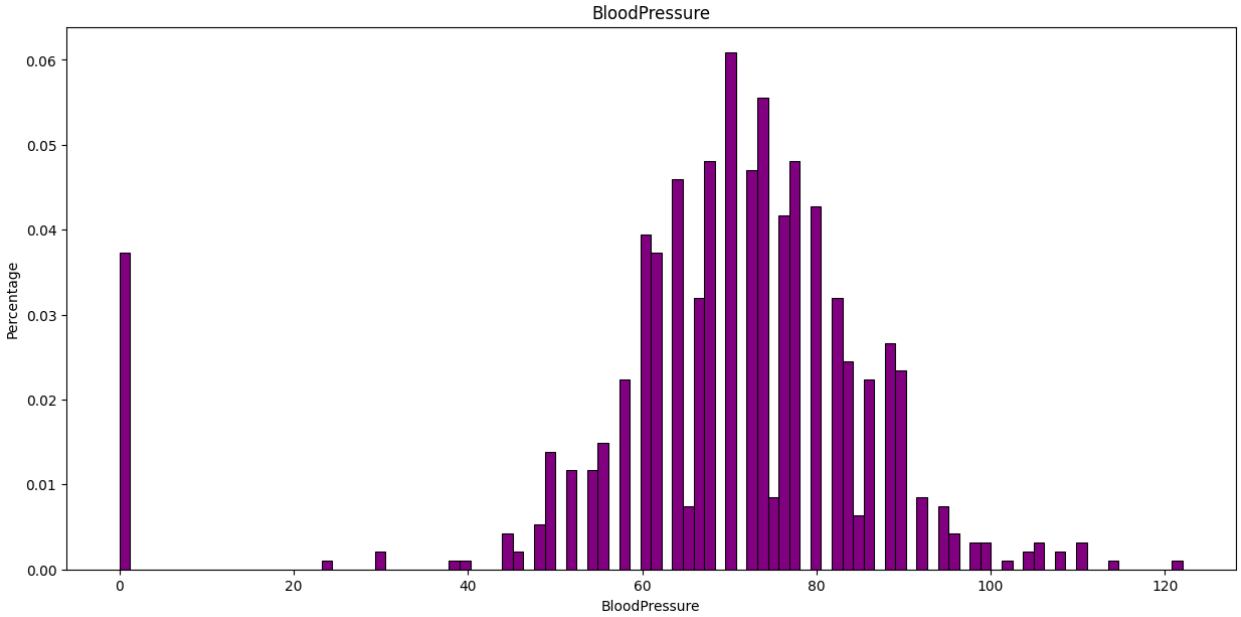
```
# Glucose
plt.figure(figsize=(15,7))
sns.histplot(diab_dataset["Glucose"], facecolor='pink', bins=100,
stat="density");
plt.ylabel("Percentage")
plt.title("Glucose")
```

```
Text(0.5, 1.0, 'Glucose')
```

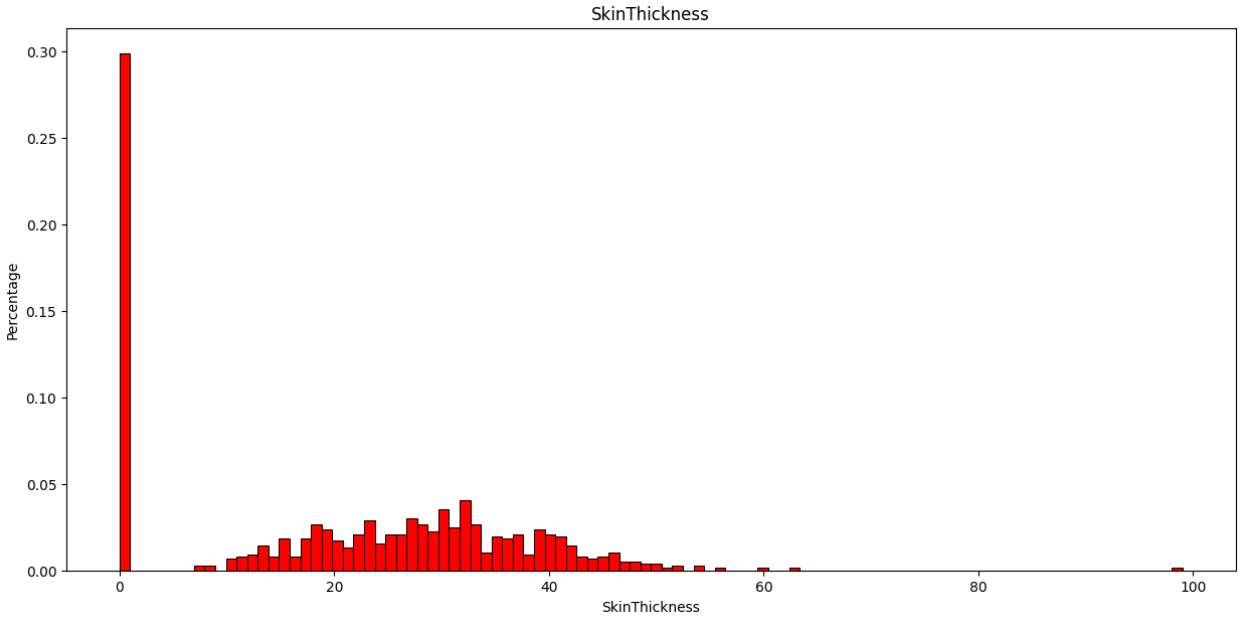



```
# BloodPressure
plt.figure(figsize=(15,7))
sns.histplot(diab_dataset["BloodPressure"], facecolor='purple',
bins=100, stat="density");
plt.ylabel("Percentage")
plt.title("BloodPressure")
```

```
Text(0.5, 1.0, 'BloodPressure')
```




```
# SkinThickness
plt.figure(figsize=(15,7))
sns.histplot(diab_dataset["SkinThickness"], facecolor='red', bins=100,
stat="density");
plt.ylabel("Percentage")
plt.title("SkinThickness")

Text(0.5, 1.0, 'SkinThickness')
```

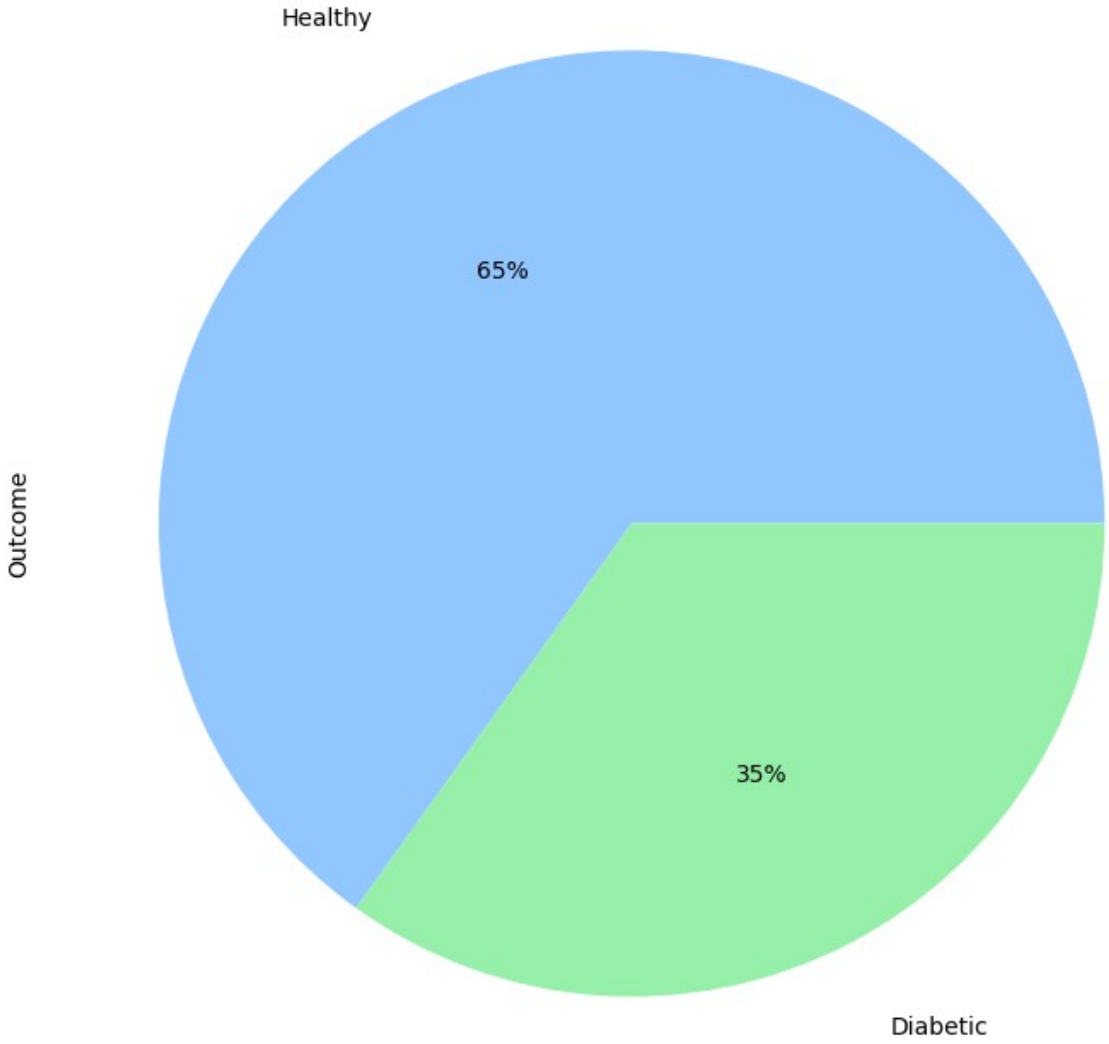



```
import matplotlib.style as style
style.available
```

```
style.use('seaborn-pastel')
labels = ["Healthy", "Diabetic"]
diab_dataset['Outcome'].value_counts().plot(kind='pie',labels=labels,
subplots=True,autopct='%1.0f%%', labeldistance=1.2,figsize=(9,9))
```

```
<ipython-input-16-149fd913f21d>:4: MatplotlibDeprecationWarning: The
seaborn styles shipped by Matplotlib are deprecated since 3.6, as they
no longer correspond to the styles shipped by seaborn. However, they
will remain available as 'seaborn-v0_8-<style>'. Alternatively,
directly use the seaborn API instead.
  style.use('seaborn-pastel')
```

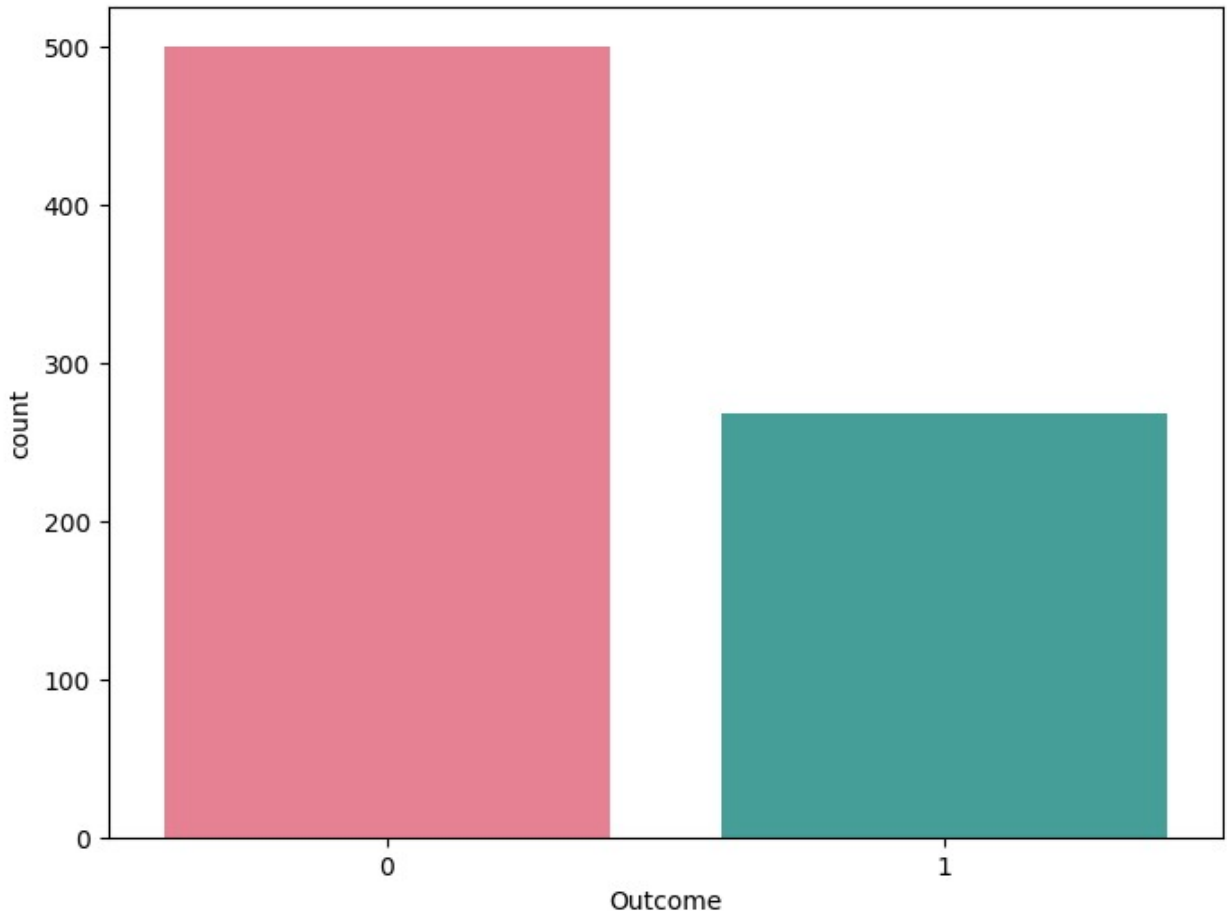
```
array([<Axes: ylabel='Outcome'>], dtype=object)
```



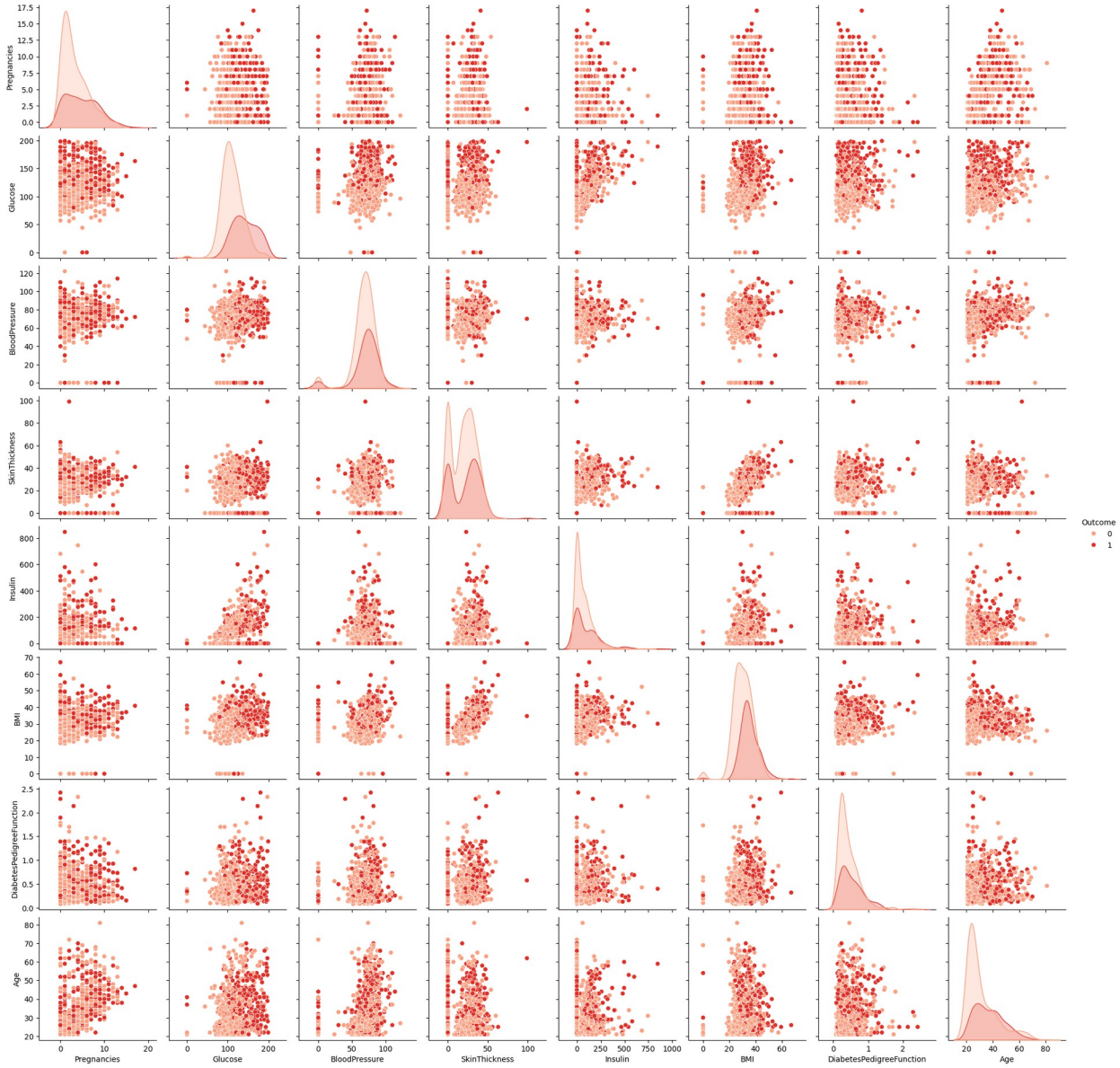

```
from matplotlib.pyplot import figure, show

figure(figsize=(8,6))
ax = sns.countplot(x=diab_dataset['Outcome'], data=diab_dataset,
palette="husl")
healthy, diabetics = diab_dataset['Outcome'].value_counts().values
print("Sample of diabetic people: ",diabetics)
print("Sample of healthy people: ",healthy)
```

```
<ipython-input-17-fb2b693c6836>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
```

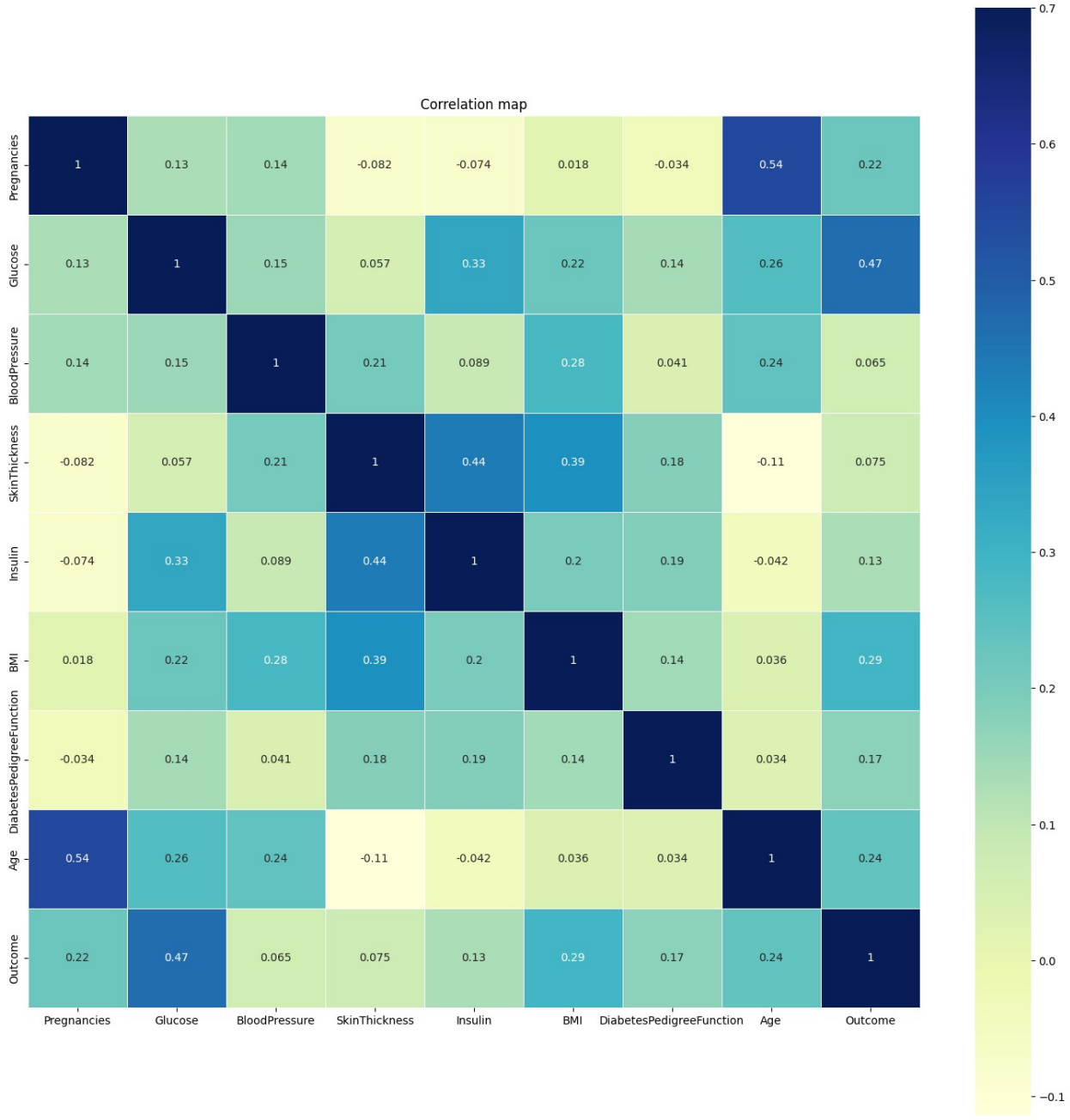
```
`legend=False` for the same effect.

  ax = sns.countplot(x=diab_dataset['Outcome'], data=diab_dataset, palette="husl")

Sample of diabetic people:  268
Sample of healthy people:  500
```




```
sns.pairplot(diab_dataset, hue="Outcome", palette="Reds")

<seaborn.axisgrid.PairGrid at 0x7c0a1457b790>
```




```
matrix = diab_dataset.corr()
f, ax = plt.subplots(figsize=(18, 18))
sns.heatmap(matrix, vmax=.7, square=True, cmap="YlGnBu", annot=True,
linewidths=.5).set_title('Correlation map')

Text(0.5, 1.0, 'Correlation map')
```




```
diab_dataset.groupby('Outcome').mean()
```


```
{"summary":"{\n  \"name\": \"diab_dataset\",\n  \"rows\": 2,\n
\"fields\": [\n    {\n      \"column\": \"Pregnancies\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.108511248584296,\n        \"min\": 3.298,\n        \"max\":
4.865671641791045,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          4.865671641791045,\n          3.298\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Glucose\",\n
```

\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
22.116505963980842,\n   \"min\": 109.98,\n   \"max\":
141.25746268656715,\n   \"num_unique_values\": 2,\n
\"samples\": [\n   141.25746268656715,\n   109.98\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"BloodPressure\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1.8672051632998017,\n   \"min\": 68.184,\n   \"max\":
70.82462686567165,\n   \"num_unique_values\": 2,\n
\"samples\": [\n   70.82462686567165,\n   68.184\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"SkinThickness\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1.7678935989570275,\n   \"min\": 19.664,\n   \"max\":
22.16417910447761,\n   \"num_unique_values\": 2,\n
\"samples\": [\n   22.16417910447761,\n   19.664\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Insulin\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
22.304849659757796,\n   \"min\": 68.792,\n   \"max\":
100.33582089552239,\n   \"num_unique_values\": 2,\n
\"samples\": [\n   100.33582089552239,\n   68.792\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"BMI\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 3.4212211239962618,\
n   \"min\": 30.3042,\n   \"max\": 35.14253731343284,\n
\"num_unique_values\": 2,\n   \"samples\": [\n
35.14253731343284,\n   30.3042\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"DiabetesPedigreeFunction\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.08539445753677459,\n   \"min\": 0.429734,\n   \"max\":
0.5505,\n   \"num_unique_values\": 2,\n   \"samples\": [\n
0.5505,\n   0.429734\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Age\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 4.155782645191446,\n   \"min\":
31.19,\n   \"max\": 37.06716417910448,\n
\"num_unique_values\": 2,\n   \"samples\": [\n
37.06716417910448,\n   31.19\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

```
# separating data and labels (drop outcome column)
X = diab_dataset.drop(columns = 'Outcome', axis=1) # dropping col,
axis=1; dropping row, axis=0
Y = diab_dataset['Outcome']

print(X)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
0              6      148             72             35        0  33.6

1              1       85             66             29        0  26.6

2              8      183             64              0        0  23.3

3              1       89             66             23       94  28.1

4              0      137             40             35      168  43.1

..           ...      ...            ...            ...      ...   ...

763           10      101             76             48      180  32.9

764            2      122             70             27        0  36.8

765            5      121             72             23      112  26.2

766            1      126             60              0        0  30.1

767            1       93             70             31        0  30.4


     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]

print(Y)

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
```

```
sc = StandardScaler()
sc.fit(X)
standardized_data = sc.transform(X)
print(standardized_data)
```

```
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
```

```
X = standardized_data
Y = diab_dataset['Outcome']
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.2, stratify = Y, random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(768, 8) (614, 8) (154, 8)
```

```
classifier  = svm.SVC(kernel='linear')
classifier.fit(X_train,Y_train)
```

```
SVC(kernel='linear')
```

```
print(sc.transform([[5,166,72,19,175,25.8,0.587,51]]))
print(classifier.predict(sc.transform([[5,166,72,19,175,25.8,0.587,51]
])))
prediction =
classifier.predict(sc.transform([[5,166,72,19,175,25.8,0.587,51]]))
if prediction[0] == 0:
  print("The person is not diabetic")
else:
  print("The person is diabetic")
```

```
[[ 0.3429808   1.41167241  0.14964075 -0.09637905  0.82661621 -
0.78595734
   0.34768723  1.51108316]]
```

```
[1]
The person is diabetic

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but StandardScaler
was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but StandardScaler
was fitted with feature names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but StandardScaler
was fitted with feature names
  warnings.warn(
```

```
Y_pred = classifier.predict(X_test)
print(np.concatenate((Y_pred.reshape(len(Y_pred),1),
Y_test.to_numpy().reshape(len(Y_test),1)),1))
```

```
[[0 0]
 [0 0]
 [1 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [1 1]
 [0 0]
 [0 1]
 [1 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 1]
 [0 0]
 [0 0]
 [1 1]
 [0 1]
 [0 0]
 [0 0]
 [0 1]
 [1 1]
 [0 0]
 [1 0]
```

```
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 1]
[0 0]
[0 1]
[0 0]
[1 1]
[0 1]
[0 0]
[1 1]
[0 0]
[0 0]
[1 1]
[0 0]
[1 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 1]
```

```
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 1]
[1 0]
[0 0]
[0 0]
[1 0]
[0 0]
[0 1]
[0 0]
[1 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[1 0]
[1 1]
[0 0]
[0 1]
[0 1]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 1]
[1 1]
[0 0]
[1 1]
[0 1]
[1 1]
[0 0]
[0 1]
[0 0]
[1 1]
[1 0]
[0 0]
[0 0]
[0 0]
```

```
 [0 1]
 [0 0]
 [0 0]
 [1 1]
 [1 1]
 [0 1]
 [0 0]
 [0 1]
 [0 1]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]]
```

```
X_train_pred = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_pred,Y_train)
print('Accuracy score of the training data: ',training_data_accuracy)
```

```
Accuracy score of the training data:  0.7866449511400652
```

```
X_test_pred = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_pred,Y_test)
print('Accuracy score of the test data: ',test_data_accuracy)
```

```
Accuracy score of the test data:  0.7727272727272727
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(Y_test, Y_pred)
print(cm)
accuracy_score(Y_test, Y_pred)
```

```
[[91  9]
 [26 28]]
```

```
0.7727272727272727
```

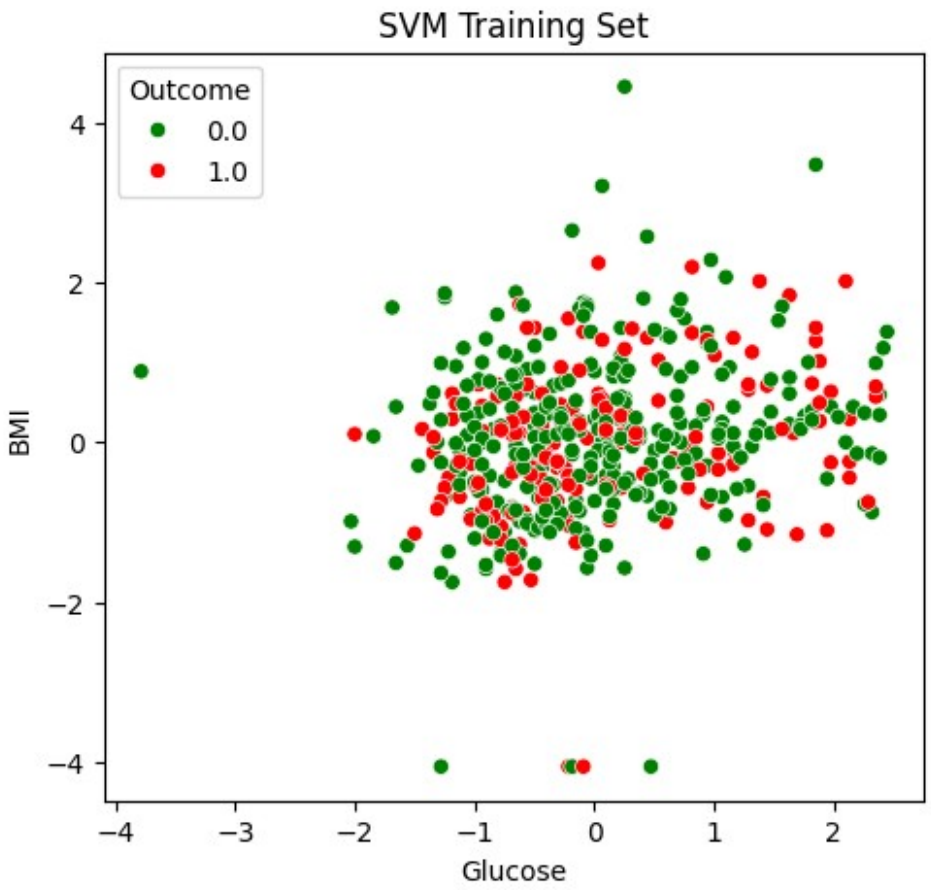
```
# Performing k-fold cross-validation
k = 5
```

```
cv_scores = cross_val_score(classifier, X_train, Y_train, cv=k)
print(f'Cross-Validation Scores: {cv_scores}')
print(f'Mean Accuracy: {np.mean(cv_scores)}')

# Performing grid search for hyperparameter tuning
param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [0.1, 0.01, 0.001,
0.0001], 'kernel': ['linear', 'rbf', 'poly']}
grid_search = GridSearchCV(estimator=classifier,
param_grid=param_grid, cv=k, scoring='accuracy')
grid_search.fit(X_train, Y_train)

# To get the best parameters and best score from grid search
best_params = grid_search.best_params_
best_score = grid_search.best_score_
print(f'Best Parameters: {best_params}')
print(f'Best Score: {best_score}')

# Predictions on test set with best parameters
best_classifier = grid_search.best_estimator_
Y_pred_test = best_classifier.predict(X_test)
test_accuracy = accuracy_score(Y_pred_test, Y_test)
print(f'Accuracy on Test Set with Best Parameters: {test_accuracy}')
```

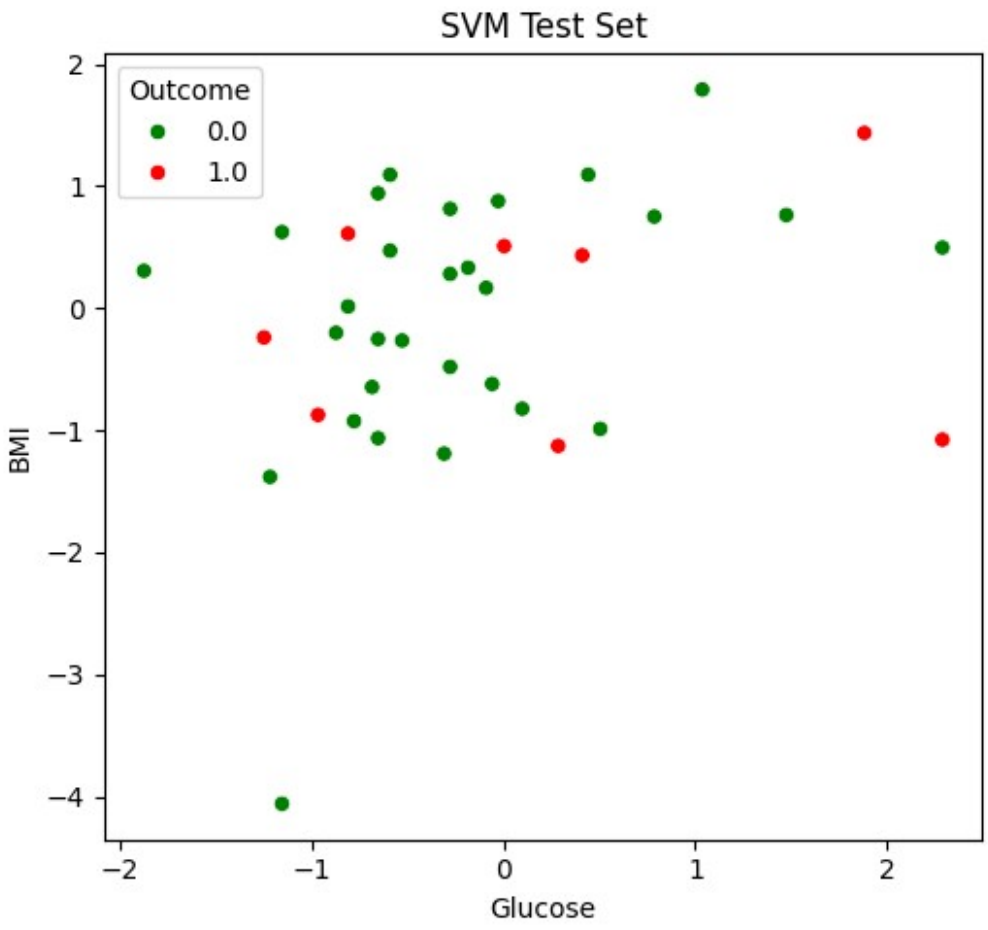
```
Cross-Validation Scores: [0.80487805 0.75609756 0.76422764 0.83739837
0.74590164]
Mean Accuracy: 0.7817006530721045
Best Parameters: {'C': 1, 'gamma': 0.1, 'kernel': 'linear'}
Best Score: 0.7817006530721045
Accuracy on Test Set with Best Parameters: 0.7727272727272727
```

```
train_df = pd.DataFrame(X_train, columns=diab_dataset.columns[:-1])
train_df['Outcome'] = Y_train
colors = ['green', 'red']
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.scatterplot(data=train_df, x='Glucose', y='BMI', hue='Outcome',
palette=colors, legend='full')
plt.title('SVM Training Set')
plt.xlabel('Glucose')
plt.ylabel('BMI')
```

```
Text(0, 0.5, 'BMI')
```




```
test_df = pd.DataFrame(X_test, columns=diab_dataset.columns[:-1])
test_df['Outcome'] = Y_test
colors = ['green', 'red']
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 2)
sns.scatterplot(data=test_df, x='Glucose', y='BMI', hue='Outcome',
palette=colors, legend='full')
plt.title('SVM Test Set')
plt.xlabel('Glucose')
plt.ylabel('BMI')

plt.tight_layout()
plt.show()
```




```
base_model = DecisionTreeClassifier(random_state=42)


base_model = DecisionTreeClassifier(random_state=42)


bagging_model = BaggingClassifier(base_estimator=base_model,
n_estimators=10, random_state=42)


bagging_model.fit(X_train, Y_train)
y_pred = bagging_model.predict(X_test)
accuracy = accuracy_score(Y_test, Y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7727272727272727
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:166:
FutureWarning: `base_estimator` was renamed to `estimator` in version
1.2 and will be removed in 1.4.
  warnings.warn(
```