In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, f1_score,
```

In [4]:
```
data = pd.read_csv(r'C:\Users\pl\Downloads\diabetes.csv')
```

In [5]:
```
data
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [6]:
```
data.drop(['Pregnancies', 'BloodPressure', 'SkinThickness'], axis=1, inplace=True
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Glucose   768 non-null    int64  
 1   Insulin   768 non-null    int64  
 2   BMI       768 non-null    float64
 3   Pedigree  768 non-null    float64
 4   Age       768 non-null    int64  
 5   Outcome   768 non-null    int64  
dtypes: float64(2), int64(4)
memory usage: 36.1 KB
```

In [8]:
```
data.head()
```

Out[8]:

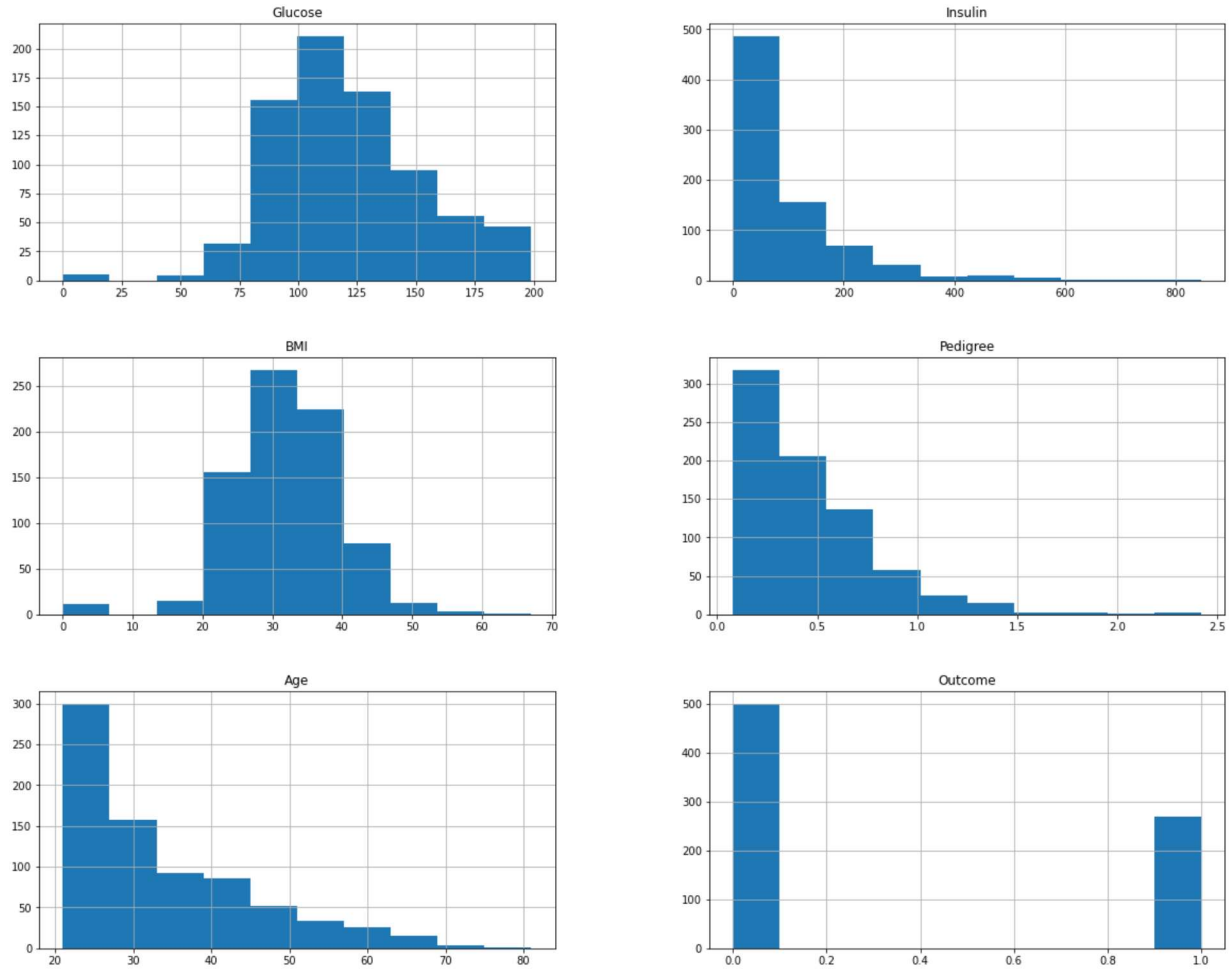
	Glucose	Insulin	BMI	Pedigree	Age	Outcome
0	148	0	33.6	0.627	50	1
1	85	0	26.6	0.351	31	0
2	183	0	23.3	0.672	32	1
3	89	94	28.1	0.167	21	0
4	137	168	43.1	2.288	33	1

In [9]:
```
data.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
Pedigree	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

In [10]:
```
hist = data.hist(figsize=(20,16))
```



In [11]:
```
target_feature = 'Outcome'
num_features = list(set(data.columns) - set([target_feature]))
```

In [12]:
```
X = data.drop(target_feature, axis=1)
y = data[target_feature]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
```

In [13]:
```
data[data['Glucose'] == 0]
```

Out[13]:

	Glucose	Insulin	BMI	Pedigree	Age	Outcome
75	0	0	24.7	0.140	22	0
182	0	23	27.7	0.299	21	0
342	0	0	32.0	0.389	22	0
349	0	0	41.0	0.346	37	1
502	0	0	39.0	0.727	41	1

In [14]:
```
data[data['BMI'] == 0]
```

Out[14]:

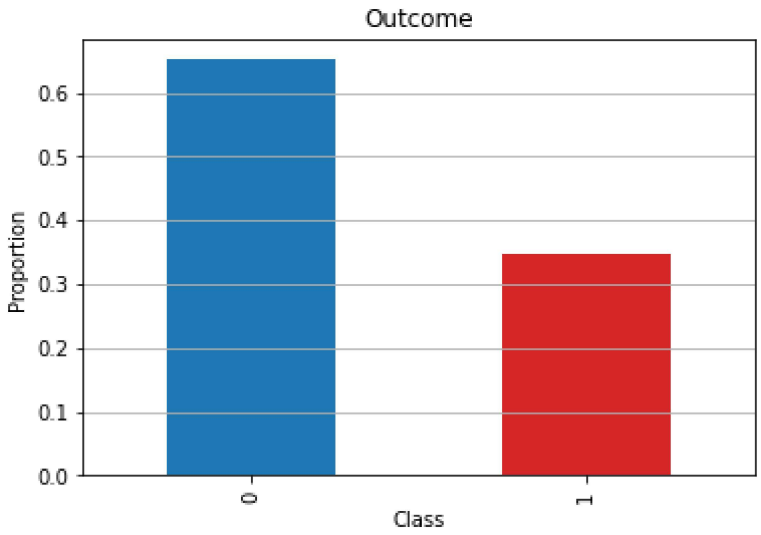
	Glucose	Insulin	BMI	Pedigree	Age	Outcome
9	125	0	0.0	0.232	54	1
49	105	0	0.0	0.305	24	0
60	84	0	0.0	0.304	21	0
81	74	0	0.0	0.102	22	0
145	102	0	0.0	0.572	21	0
371	118	89	0.0	1.731	21	0
426	94	0	0.0	0.256	25	0
494	80	0	0.0	0.174	22	0
522	114	0	0.0	0.189	26	0
684	136	0	0.0	0.640	69	0
706	115	0	0.0	0.261	30	1

In [15]:
```
imputer = SimpleImputer(missing_values=0.0, strategy='median')
imputer.fit(X_train[['Glucose','BMI']])
X_train[['Glucose','BMI']] = imputer.transform(X_train[['Glucose','BMI']])
X_test[['Glucose','BMI']] = imputer.transform(X_test[['Glucose','BMI']])
```

```
C:\Users\pl\anaconda3\lib\site-packages\pandas\core\frame.py:3678: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  self[col] = igetitem(value, i)
C:\Users\pl\anaconda3\lib\site-packages\pandas\core\frame.py:3678: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  self[col] = igetitem(value, i)
```

In [16]:
```
y_train.value_counts(normalize=True).plot.bar(color=['tab:blue', 'tab:red'])
plt.grid(axis='y')
plt.title(target_feature)
plt.xlabel('Class')
plt.ylabel('Proportion')
```

Out[16]: Text(0, 0.5, 'Proportion')



In [21]:
```
X_train
```

Out[21]:

	Glucose	Insulin	BMI	Pedigree	Age
22	196.0	0	39.8	0.451	41
497	81.0	76	30.1	0.547	25
395	127.0	275	27.7	1.600	25
381	105.0	0	20.0	0.236	22
258	193.0	375	25.9	0.655	24
...	...	...	...	...	...
456	135.0	0	26.7	0.687	62
435	141.0	0	42.4	0.205	29
398	82.0	0	21.1	0.389	25
48	103.0	0	39.1	0.344	31
294	161.0	0	21.9	0.254	65

614 rows × 5 columns

In [22]:
```
X_test
```

Out[22]:

	Glucose	Insulin	BMI	Pedigree	Age
680	56.0	45	24.2	0.332	22
607	92.0	41	19.5	0.482	25
639	100.0	46	19.5	0.149	28
638	97.0	91	40.9	0.871	32
295	151.0	120	35.5	0.692	28
...	...	...	...	...	...
526	97.0	82	18.2	0.299	21
685	129.0	205	33.2	0.591	25
391	166.0	0	45.7	0.340	27
654	106.0	135	34.2	0.142	22
315	112.0	94	34.1	0.315	26

154 rows × 5 columns

In [23]:
```
y_train
```

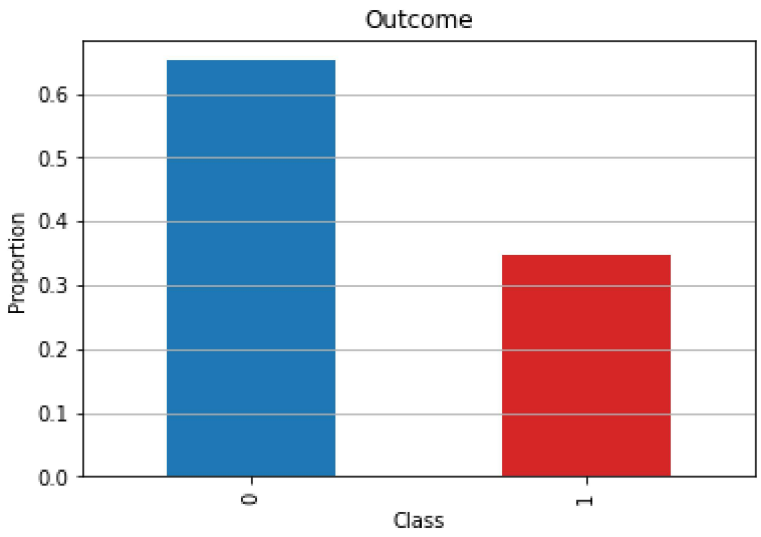
Out[23]:
```
22     1
497    0
395    0
381    0
258    0
      ..
456    0
435    1
398    0
48     1
294    0
Name: Outcome, Length: 614, dtype: int64
```

In [24]:
```
y_test
```

Out[24]:
```
680    0
607    0
639    0
638    1
295    0
      ..
526    0
685    0
391    1
654    0
315    0
Name: Outcome, Length: 154, dtype: int64
```

In [25]:
```
y_train.value_counts(normalize=True).plot.bar(color=['tab:blue', 'tab:red'])
plt.grid(axis='y')
plt.title(target_feature)
plt.xlabel('Class')
plt.ylabel('Proportion')
```

Out[25]:
```
Text(0, 0.5, 'Proportion')
```



In [26]:
```
X_train[num_features].corr().style.background_gradient(cmap='coolwarm')
```

Out[26]:

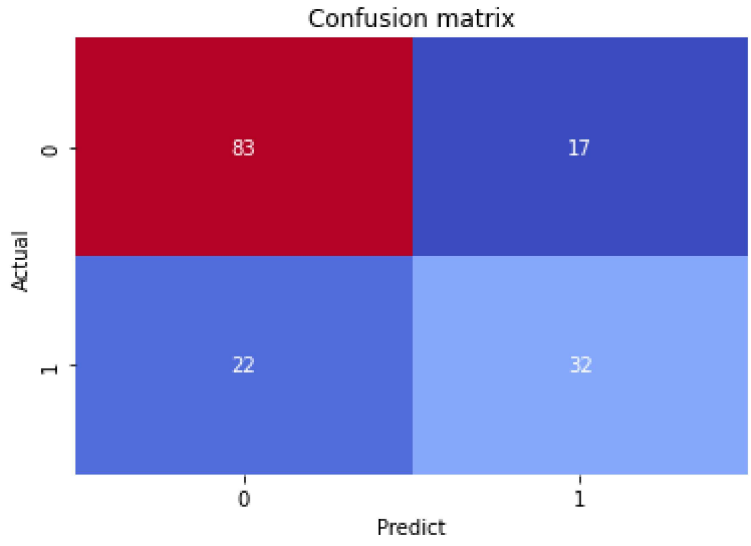
	Glucose	Insulin	Pedigree	BMI	Age
Glucose	1.000000	0.321751	0.145176	0.223262	0.263797
Insulin	0.321751	1.000000	0.207136	0.192143	-0.026828
Pedigree	0.145176	0.207136	1.000000	0.165553	0.018448
BMI	0.223262	0.192143	0.165553	1.000000	0.006558
Age	0.263797	-0.026828	0.018448	0.006558	1.000000

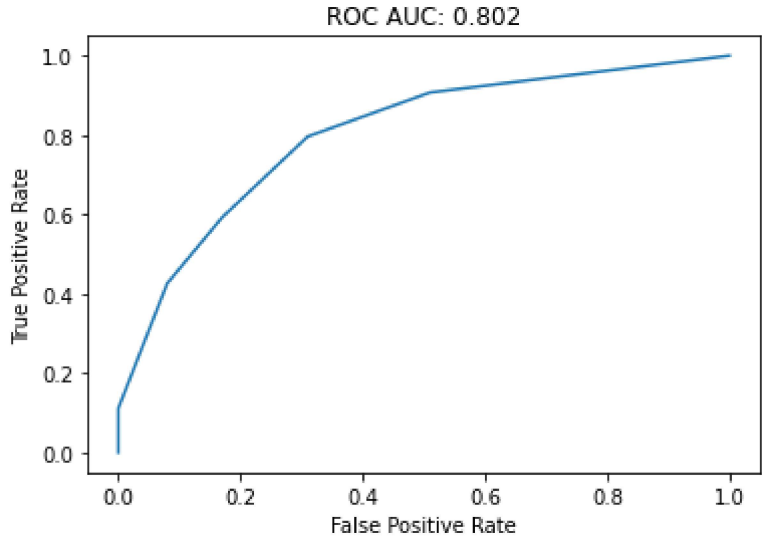
In [28]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_train = pd.DataFrame(X_train, columns=num_features)
X_test = scaler.transform(X_test)
X_test = pd.DataFrame(X_test, columns=num_features)
```

In [29]:
```
def print_metrics(y_true, y_pred):
    print('Metrics:')
    print(f'f1_score = {f1_score(y_true=y_true, y_pred=y_pred).round(3)}')
    print(f'recall_score = {recall_score(y_true=y_true, y_pred=y_pred).round(3)}'
    print(f'precision_score = {precision_score(y_true=y_true, y_pred=y_pred).roun

def print_confusion_matrix(y_true, y_pred):
    sns.heatmap(confusion_matrix(y_true=y_true, y_pred=y_pred), annot=True, cmap=
    plt.title('Confusion matrix')
    plt.xlabel('Predict')
    plt.ylabel('Actual')
    plt.show()

def print_roc_auc(y_true, y_pred_prob):
    fpr, tpr, thresholds = roc_curve(y_true, y_pred_prob)
    auc = roc_auc_score(y_true, y_pred_prob)

    plt.plot(fpr, tpr)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC AUC: {auc.round(3)}')
    plt.show()

def print_params(model):
    print('Model parameters:')
    print(f'K neighbors = {model.n_neighbors}')
    print(f'Power = {model.p}')

def start_train(model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)
    y_prep = model.predict(X_test)
    y_pred_prob = model.predict_proba(X_test)[:,1]

    print_params(model=model)
    print_metrics(y_true=y_test, y_pred=y_prep)
    print_confusion_matrix(y_true=y_test, y_pred=y_prep)
    print_roc_auc(y_true=y_test, y_pred_prob=y_pred_prob)
```

In [30]:
```
start_train(model=KNeighborsClassifier(), X_train=X_train, y_train=y_train, X_tes
```

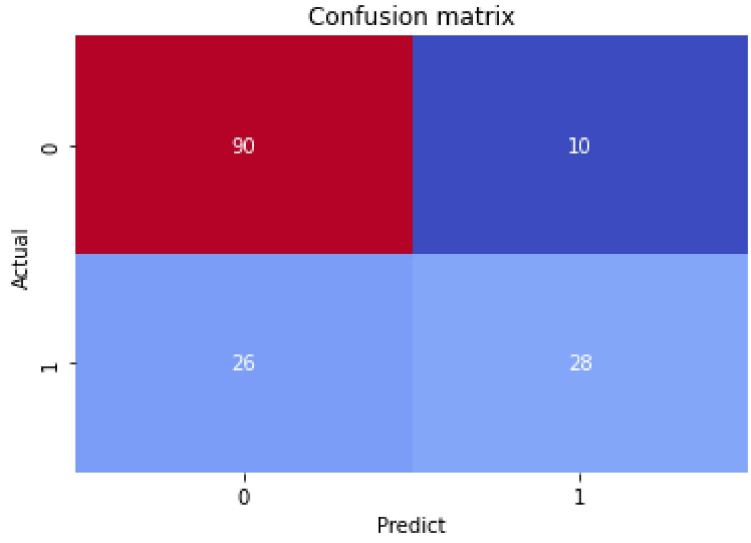
```
Model parameters:
K neighbors = 5
Power = 2
Metrics:
f1_score = 0.621
recall_score = 0.593
precision_score = 0.653
```

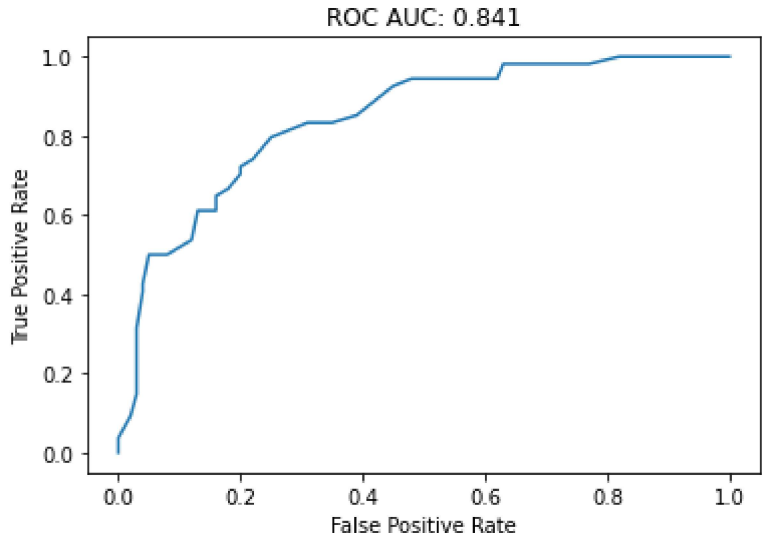




In [32]:
```
param_grid = {
    'n_neighbors': range(1, 51),
    'p': range(1, 4)
}
grid = GridSearchCV(estimator=KNeighborsClassifier(), param_grid=param_grid, cv=5
grid.fit(X_train, y_train)
model = grid.best_estimator_
start_train(model=model, X_train=X_train, y_train=y_train, X_test=X_test, y_test=
```

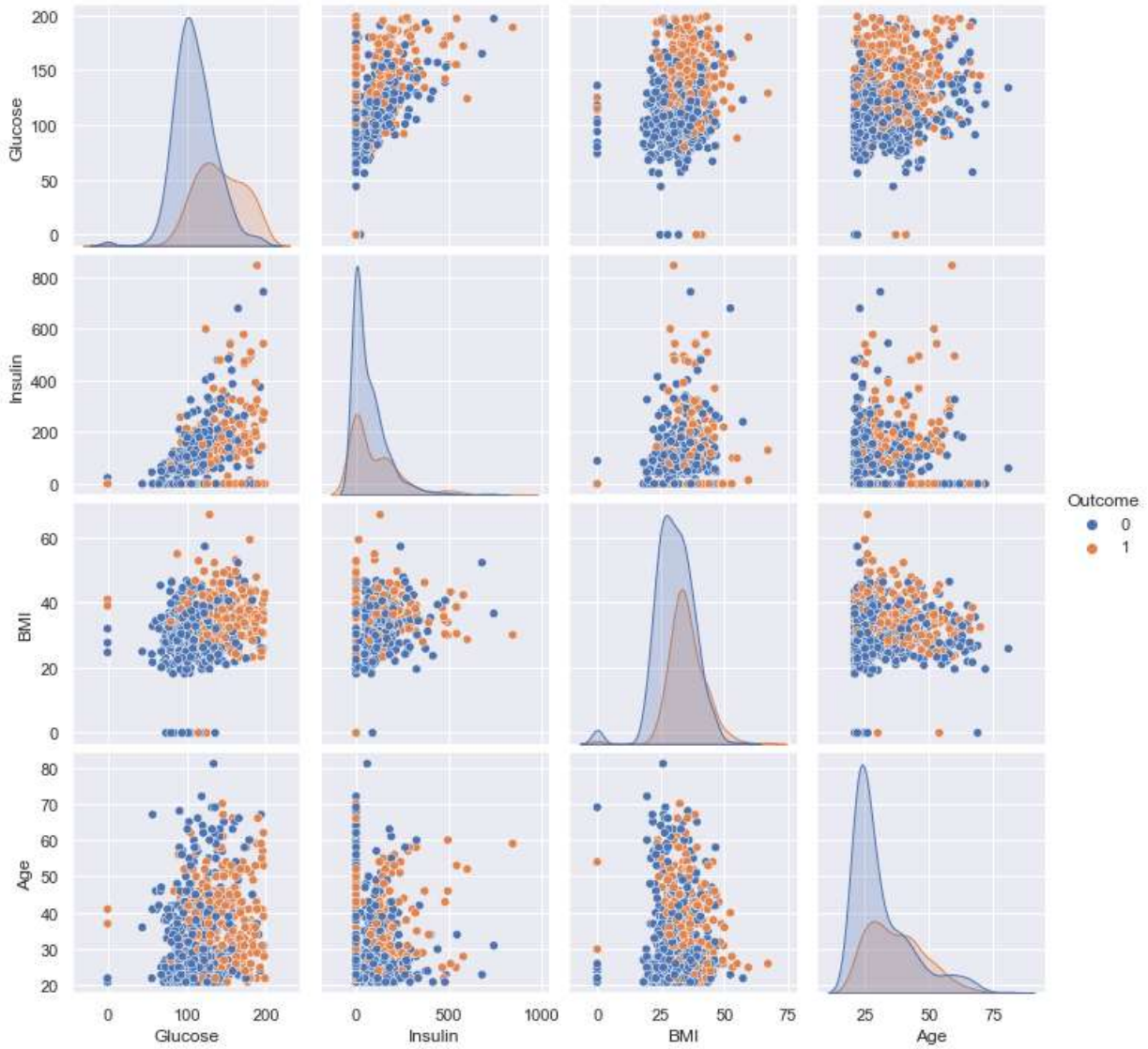
```
Model parameters:
K neighbors = 43
Power = 1
Metrics:
f1_score = 0.609
recall_score = 0.519
precision_score = 0.737
```





In [34]:
```
graph = ['Glucose','Insulin','BMI','Age','Outcome']
sns.set()
print(sns.pairplot(data[graph],hue='Outcome', diag_kind='kde'))
```

```
<seaborn.axisgrid.PairGrid object at 0x0000020553803AC0>
```



In [ ]: