In [2]:

```
import numpy as np
import pandas as pd
import seaborn as sns
%matplotlib inline
```

In [3]:

```
df=pd.read_csv("health care diabetes.csv")
```

In [5]:

```
df.head()
```

Out[5]:

	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 [6]:

```
df.describe()
```

Out[6]:

	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 [7]:

```
df.isna().sum()
```

Out[7]:

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

In [13]:

```
m=df.median()
m
```

Out[13]:

```
Pregnancies                 3.0000
Glucose                   117.0000
BloodPressure              72.0000
SkinThickness              23.0000
Insulin                    30.5000
BMI                        32.0000
DiabetesPedigreeFunction    0.3725
Age                        29.0000
Outcome                     0.0000
dtype: float64
```

In [19]:

```
df['Insulin'].replace(0,df['Insulin'].median(),inplace=True)
```

In [20]:

```
df.head(10)
```

Out[20]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	30.5	33.6	0.627	50	1
1	1	85	66	29	30.5	26.6	0.351	31	0
2	8	183	64	0	30.5	23.3	0.672	32	1
3	1	89	66	23	94.0	28.1	0.167	21	0
4	0	137	40	35	168.0	43.1	2.288	33	1
5	5	116	74	0	30.5	25.6	0.201	30	0
6	3	78	50	32	88.0	31.0	0.248	26	1
7	10	115	0	0	30.5	35.3	0.134	29	0
8	2	197	70	45	543.0	30.5	0.158	53	1
9	8	125	96	0	30.5	0.0	0.232	54	1

In [23]:

```
df['Pregnancies'].replace(0,df['Pregnancies'].median(),inplace=True)
df['Glucose'].replace(0,df['Glucose'].median(),inplace=True)
df['BloodPressure'].replace(0,df['BloodPressure'].median(),inplace=True)
df['SkinThickness'].replace(0,df['SkinThickness'].median(),inplace=True)
df['BMI'].replace(0,df['BMI'].median(),inplace=True)
df['DiabetesPedigreeFunction'].replace(0,df['DiabetesPedigreeFunction'].median(),inplace=True)
df['Age'].replace(0,df['Age'].median(),inplace=True)
```

In [24]:

```
df.head(10)
```

Out[24]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	30.5	33.6	0.627	50	1
1	1	85	66	29	30.5	26.6	0.351	31	0
2	8	183	64	23	30.5	23.3	0.672	32	1
3	1	89	66	23	94.0	28.1	0.167	21	0
4	3	137	40	35	168.0	43.1	2.288	33	1
5	5	116	74	23	30.5	25.6	0.201	30	0
6	3	78	50	32	88.0	31.0	0.248	26	1
7	10	115	72	23	30.5	35.3	0.134	29	0
8	2	197	70	45	543.0	30.5	0.158	53	1
9	8	125	96	23	30.5	32.0	0.232	54	1

In [41]:

```
df.dtypes
```

Out[41]:

```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                   float64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

In [25]:

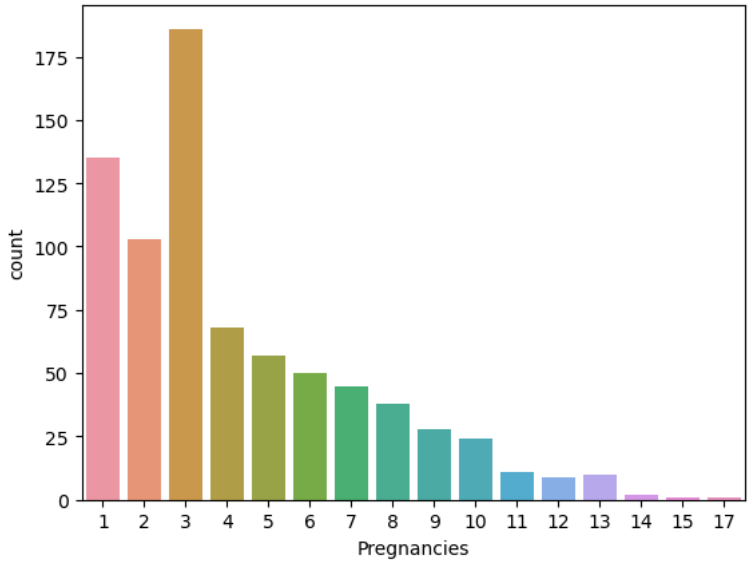
```
import matplotlib.pyplot as plt
```

In [46]:

```
sns.countplot(x=df['Pregnancies'])
```

Out[46]:

```
<AxesSubplot:xlabel='Pregnancies', ylabel='count'>
```

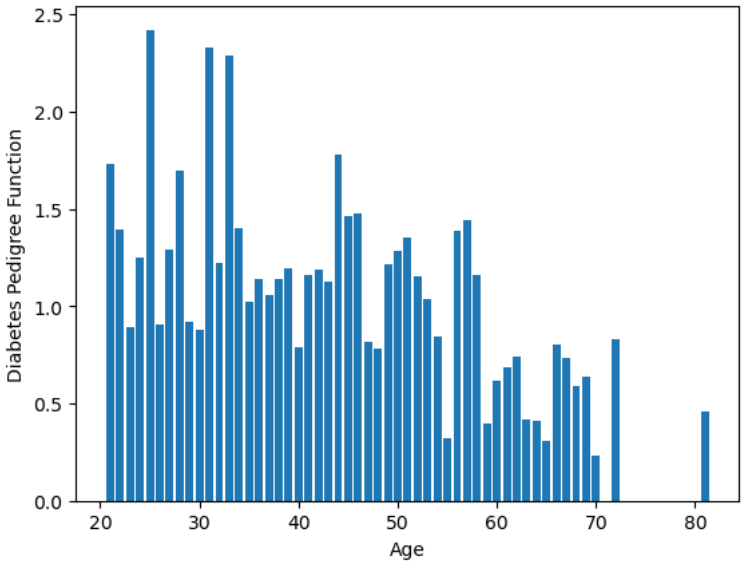


In [68]:

```
plt.bar(df['Age'],df['DiabetesPedigreeFunction'])
plt.xlabel("Age")
plt.ylabel("Diabetes Pedigree Function")
```

Out[68]:

```
Text(0, 0.5, 'Diabetes Pedigree Function')
```



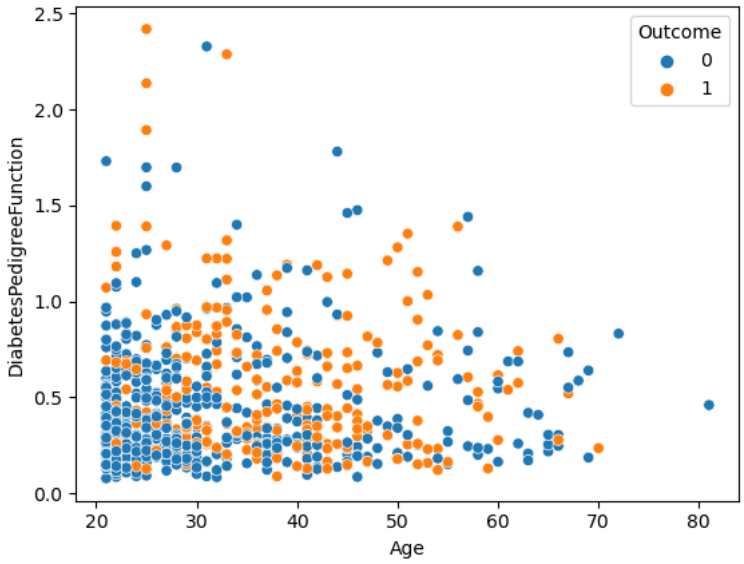
In [77]:

```
sns.scatterplot(df['Age'],df['DiabetesPedigreeFunction'],hue=df['Outcome'])
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword
args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[77]:

```
<AxesSubplot:xlabel='Age', ylabel='DiabetesPedigreeFunction'>
```



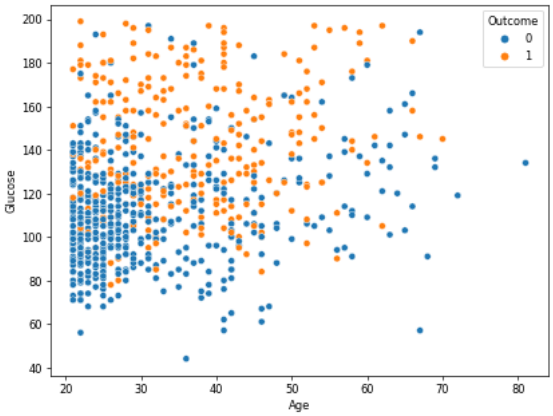
In [85]:

```
plt.figure(figsize=(8,6),dpi=60)
sns.scatterplot(df['Age'],df['Glucose'],hue=df['Outcome'])
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword
args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[85]:

```
<AxesSubplot:xlabel='Age', ylabel='Glucose'>
```



In [91]:

```
dtype_counts=df.dtypes.value_counts()
dtype_counts
```

Out[91]:

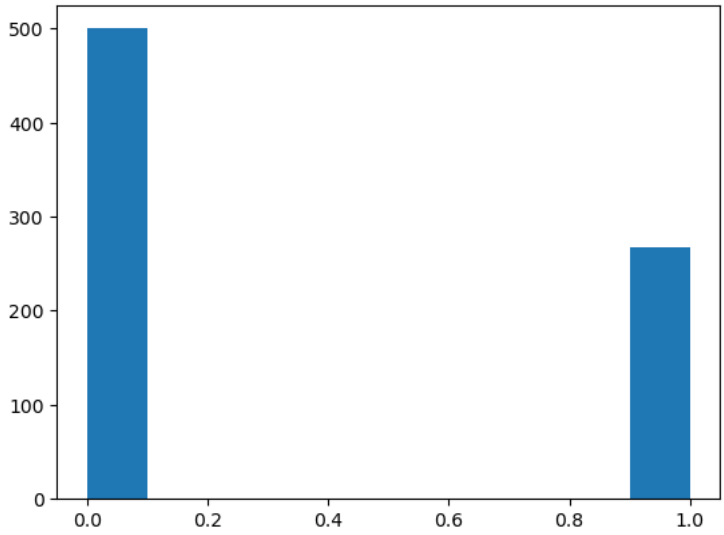
```
int64      6
float64    3
dtype: int64
```

In [97]:

```
plt.hist(df['Outcome'])
```

Out[97]:

```
(array([500.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 268.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



In [100]:

```
df['Outcome'].value_counts()
```

Out[100]:

```
0    500
1    268
Name: Outcome, dtype: int64
```

In [103]:

```
pip install imblearn
```

```
Collecting imblearn
  Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Collecting imbalanced-learn
  Downloading imbalanced_learn-0.10.1-py3-none-any.whl (226 kB)
     ---------------------------------- 226.0/226.0 kB 234.0 kB/s eta 0:00:00
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\dell\anaconda3\lib\site-packages (from imbalanced-learn->imb
learn) (1.0.2)
Requirement already satisfied: scipy>=1.3.2 in c:\users\dell\anaconda3\lib\site-packages (from imbalanced-learn->imblearn)
(1.9.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\dell\anaconda3\lib\site-packages (from imbalanced-learn->im
blearn) (2.2.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\dell\anaconda3\lib\site-packages (from imbalanced-learn->imblearn)
(1.21.5)
Collecting joblib>=1.1.1
  Downloading joblib-1.2.0-py3-none-any.whl (297 kB)
     ---------------------------------- 298.0/298.0 kB 259.4 kB/s eta 0:00:00
Installing collected packages: joblib, imbalanced-learn, imblearn
  Attempting uninstall: joblib
    Found existing installation: joblib 1.1.0
    Uninstalling joblib-1.1.0:
      Successfully uninstalled joblib-1.1.0
Successfully installed imbalanced-learn-0.10.1 imblearn-0.0 joblib-1.2.0
Note: you may need to restart the kernel to use updated packages.
```

In [107]:

```
from imblearn import under_sampling, over_sampling
from imblearn.over_sampling import SMOTE
```

In [140]:

```
X=df.iloc[:,:-1] # All rows and all columns except last column
y=df.iloc[:,-1] # All rows of the last column
```

In [141]:

```
sm=SMOTE(sampling_strategy='minority',random_state=0)
X_train_res, y_train_res = sm.fit_resample(X_train,y_train)
```

In [142]:

```
df['Outcome'].value_counts()
```

Out[142]:

```
0    500
1    268
Name: Outcome, dtype: int64
```

In [58]:

```
## Further plotting has been done in Tableau
df.corr()
```

Out[58]:

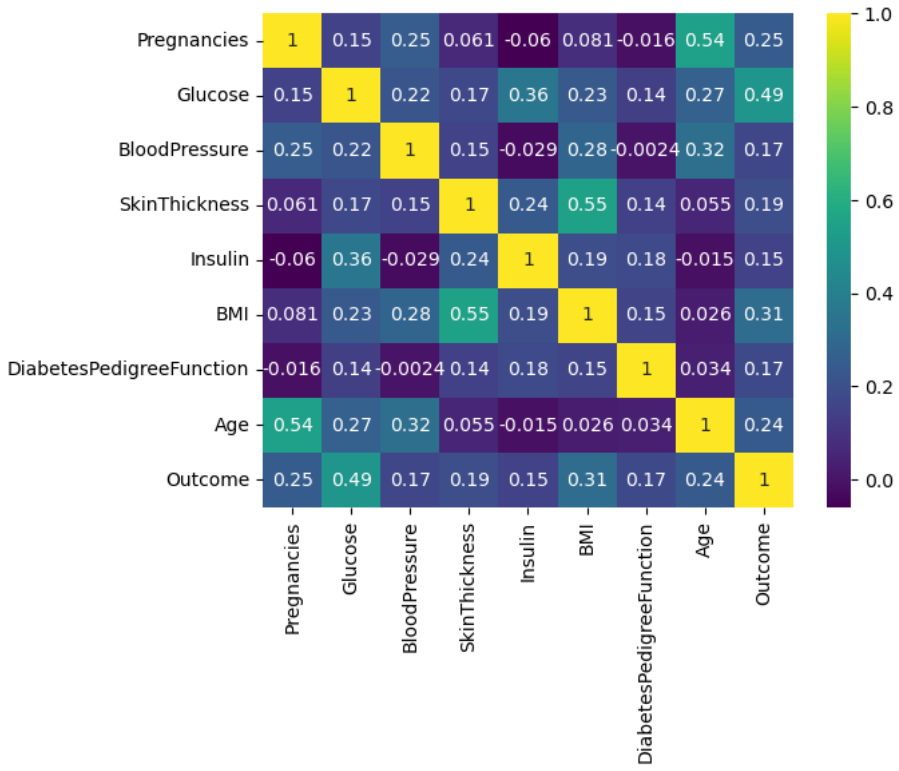
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.149326	0.247530	0.060706	-0.059580	0.080540	-0.016151	0.538169	0.245466
Glucose	0.149326	1.000000	0.218937	0.172143	0.357573	0.231400	0.137327	0.266909	0.492782
BloodPressure	0.247530	0.218937	1.000000	0.147809	-0.028721	0.281132	-0.002378	0.324915	0.165723
SkinThickness	0.060706	0.172143	0.147809	1.000000	0.238188	0.546951	0.142977	0.054514	0.189065
Insulin	-0.059580	0.357573	-0.028721	0.238188	1.000000	0.189022	0.178029	-0.015413	0.148457
BMI	0.080540	0.231400	0.281132	0.546951	0.189022	1.000000	0.153506	0.025744	0.312249
DiabetesPedigreeFunction	-0.016151	0.137327	-0.002378	0.142977	0.178029	0.153506	1.000000	0.033561	0.173844
Age	0.538169	0.266909	0.324915	0.054514	-0.015413	0.025744	0.033561	1.000000	0.238356
Outcome	0.245466	0.492782	0.165723	0.189065	0.148457	0.312249	0.173844	0.238356	1.000000

In [61]:

```
sns.heatmap(df.corr(),cmap='viridis',annot=True)
```

Out[61]:

```
<AxesSubplot:>
```

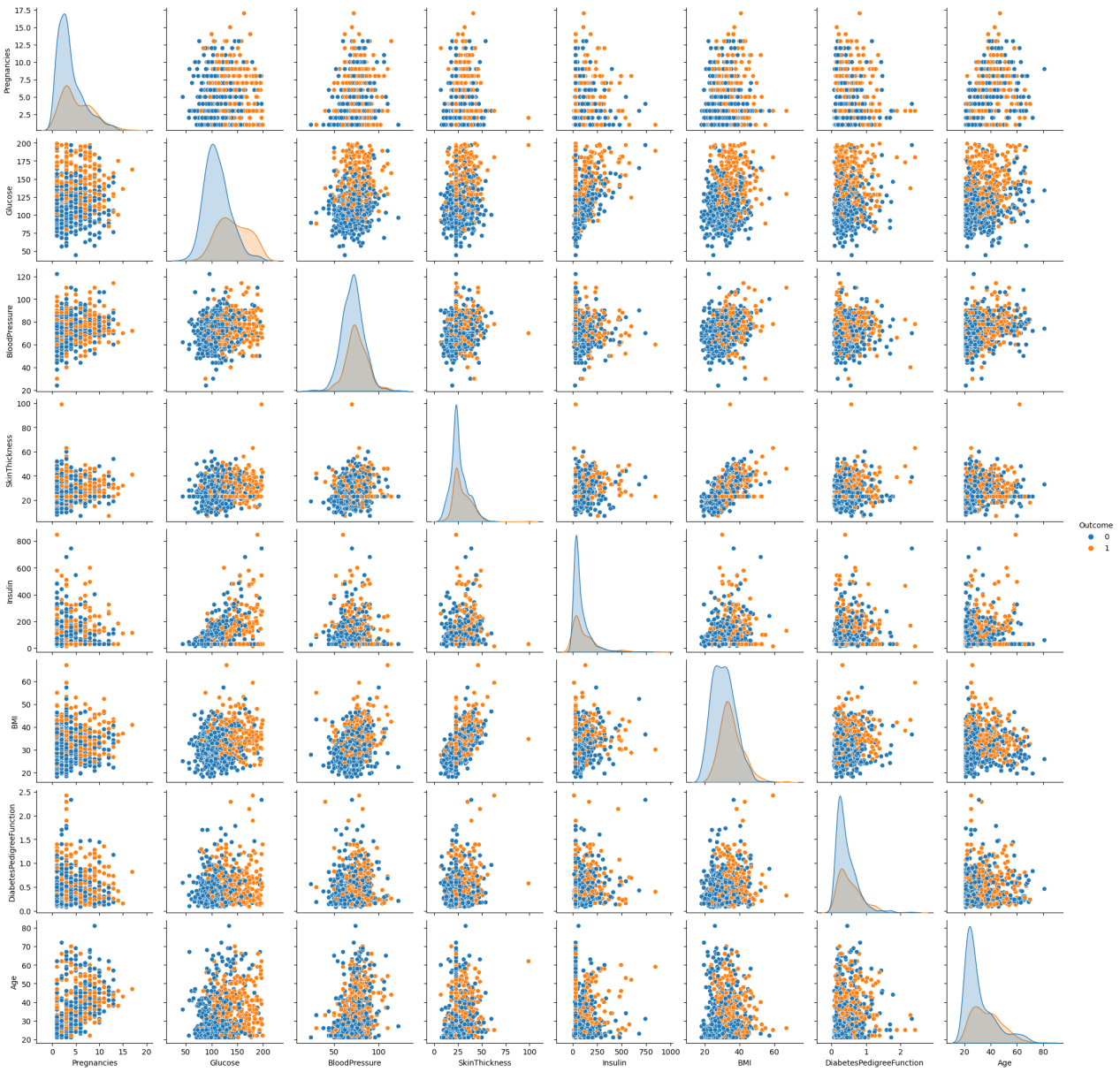


In [62]:

```
sns.pairplot(df,hue='Outcome')
```

Out[62]:

```
<seaborn.axisgrid.PairGrid at 0x1d55312f9a0>
```



In [90]:

```
df.to_csv("health care diabetes new.csv",index=False)
```

In [109]:

```
np.abs(df.corr()['Outcome']).sort_values().tail(6)
```

Out[109]:

```
SkinThickness    0.189065
Age              0.238356
Pregnancies      0.245466
BMI              0.312249
Glucose          0.492782
Outcome          1.000000
Name: Outcome, dtype: float64
```

In [144]:

```
from sklearn.model_selection import train_test_split
```

In [145]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

In [146]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[146]:

```
((537, 8), (231, 8), (537,), (231,))
```

In [150]:

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train,y_train)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to conver
ge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/
linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[150]:

```
LogisticRegression()
```

In [151]:

```
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.776536312849162
0.7662337662337663
```

In [154]:

```
from sklearn.metrics import classification_report,confusion_matrix,plot_confusion_matrix
```

In [157]:

```
cm=confusion_matrix(y,model.predict(X))
cm
```

Out[157]:

```
array([[442,  58],
       [116, 152]], dtype=int64)
```

In [158]:

```
print(classification_report(y,model.predict(X)))
```

```
              precision    recall  f1-score   support

           0       0.79      0.88      0.84       500
           1       0.72      0.57      0.64       268

    accuracy                           0.77       768
   macro avg       0.76      0.73      0.74       768
weighted avg       0.77      0.77      0.77       768
```

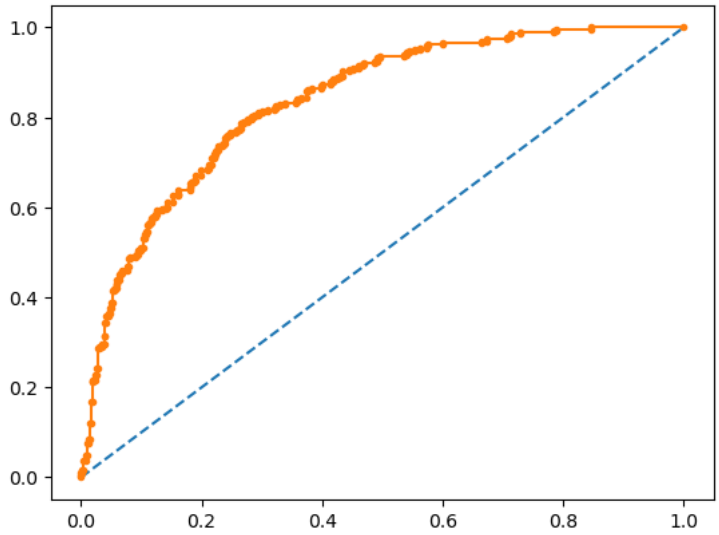
In [159]:

```
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

# predict probabilities
probs = model.predict_proba(X)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(y, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(y, probs)
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
```

```
AUC: 0.835
```

Out[159]:

```
[<matplotlib.lines.Line2D at 0x1d56242f9d0>]
```



In [161]:

```
#Applying Decission Tree Classifier
from sklearn.tree import DecisionTreeClassifier
model2 = DecisionTreeClassifier(max_depth=5)
model2.fit(X_train,y_train)
```

Out[161]:

```
DecisionTreeClassifier(max_depth=5)
```

In [162]:

```
model2.score(X_train,y_train)
```

Out[162]:

```
0.8324022346368715
```

In [163]:

```
model2.score(X_test,y_test)
```

Out[163]:

```
0.7748917748917749
```

In [164]:

```
#Applying Random Forest
from sklearn.ensemble import RandomForestClassifier
model3 = RandomForestClassifier(n_estimators=11)
model3.fit(X_train,y_train)
```

Out[164]:

```
RandomForestClassifier(n_estimators=11)
```

In [165]:

```
model3.score(X_train,y_train)
```

Out[165]:

```
0.9906890130353817
```

In [166]:

```
model3.score(X_test,y_test)
```

Out[166]:

```
0.7445887445887446
```

In [167]:

```
#Support Vector Classifier

from sklearn.svm import SVC
model4 = SVC(kernel='rbf', gamma='auto')
model4.fit(X_train,y_train)
```

Out[167]:

```
SVC(gamma='auto')
```

In [170]:

```
model4.score(X_train,y_train)
```

Out[170]:

```
1.0
```

In [171]:

```
model4.score(X_test,y_test)
```

Out[171]:

```
0.6796536796536796
```

In [172]:

```
#Applying K-NN
from sklearn.neighbors import KNeighborsClassifier
model5 = KNeighborsClassifier(n_neighbors=7, metric='minkowski',p = 2)
model5.fit(X_train,y_train)
```

Out[172]:

```
KNeighborsClassifier(n_neighbors=7)
```

In [173]:

```
model5.score(X_train,y_train)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction f
unctions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.1
1.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is t
aken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warni
ng.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Out[173]:

```
0.7858472998137802
```

In [174]:

```
model5.score(X_test,y_test)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction f
unctions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.1
1.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is t
aken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warni
ng.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Out[174]:

```
0.7316017316017316
```

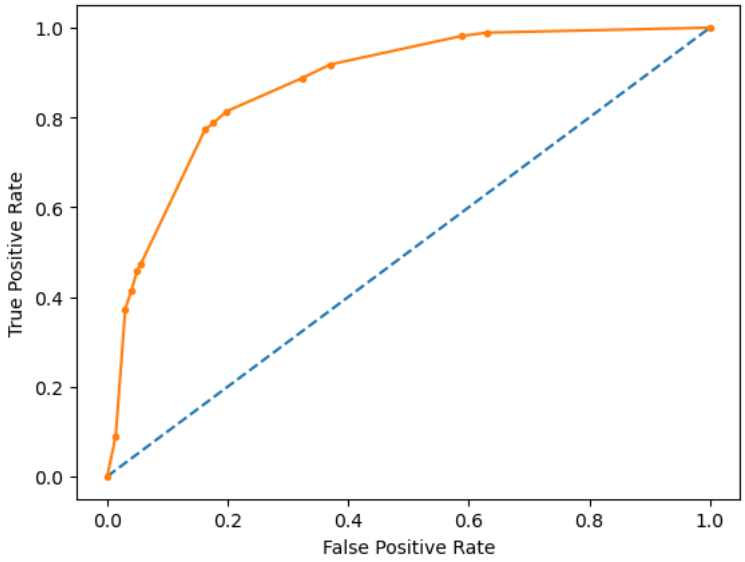
In [175]:

```
#Preparing ROC Curve (Receiver Operating Characteristics Curve) Decission Tree Classifier


from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

# predict probabilities
probs = model2.predict_proba(X)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(y, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(y, probs)
print("True Positive Rate - {}, False Positive Rate - {} Thresholds - {}".format(tpr,fpr,thresholds))
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

```
AUC: 0.875
True Positive Rate - [0.         0.08955224 0.37313433 0.4141791  0.45895522 0.4738806
 0.77238806 0.78731343 0.81343284 0.8880597  0.91791045 0.98134328
 0.98880597 1.        ], False Positive Rate - [0.    0.014 0.03  0.04  0.05  0.056 0.162 0.176 0.198 0.324 0.37  0.588
 0.63  1.   ] Thresholds - [2.         1.         0.93548387 0.76923077 0.7        0.66666667
 0.58252427 0.5        0.33333333 0.22       0.2        0.11111111
 0.08333333 0.        ]
```

Out[175]:

```
Text(0, 0.5, 'True Positive Rate')
```



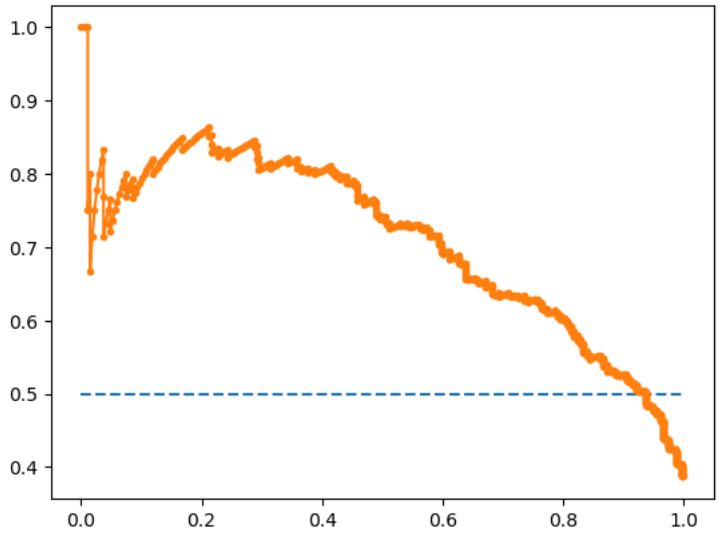
In [179]:

```
#Precision Recall Curve for Logistic Regression

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model.predict_proba(X)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model.predict(X)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y, probs)
# calculate F1 score
f1 = f1_score(y, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(y, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.636 auc=0.709 ap=0.710
```

Out[179]:

```
[<matplotlib.lines.Line2D at 0x1d562c13b80>]
```



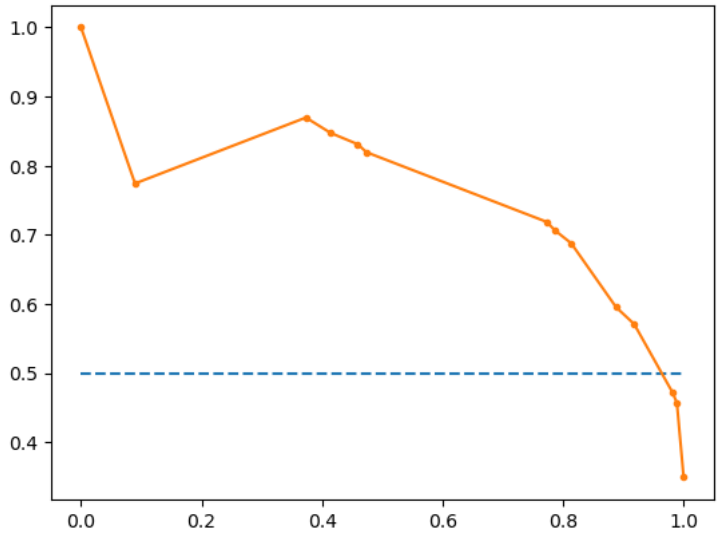
In [182]:

```
#Precision Recall Curve for Decission Tree Classifier


from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model2.predict_proba(X)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model2.predict(X)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y, probs)
# calculate F1 score
f1 = f1_score(y, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(y, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.745 auc=0.762 ap=0.742
```

Out[182]:

```
[<matplotlib.lines.Line2D at 0x1d562b4ca30>]
```



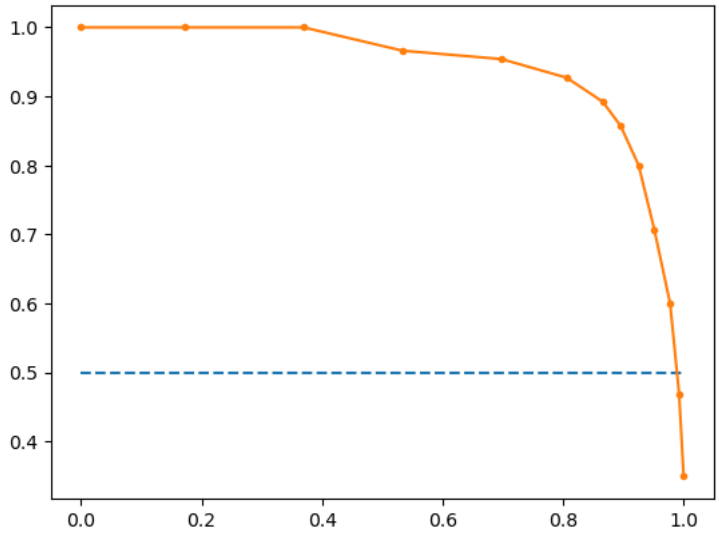
In [183]:

```
#Precision Recall Curve for Random Forest

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model3.predict_proba(X)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model3.predict(X)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y, probs)
# calculate F1 score
f1 = f1_score(y, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(y, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.879 auc=0.943 ap=0.931
```

Out[183]:

```
[<matplotlib.lines.Line2D at 0x1d562bb7c40>]
```



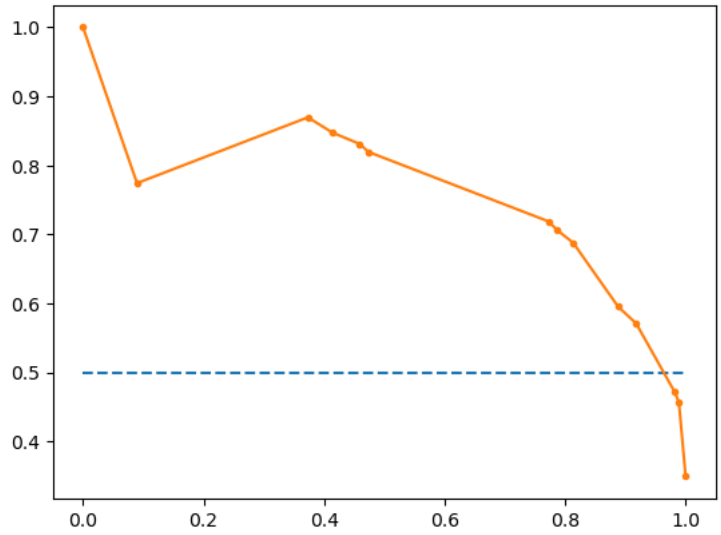
In [184]:

```
#Precision Recall Curve for KNN

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model2.predict_proba(X)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model2.predict(X)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(y, probs)
# calculate F1 score
f1 = f1_score(y, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(y, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.745 auc=0.762 ap=0.742
```

Out[184]:

```
[<matplotlib.lines.Line2D at 0x1d5627bb4c0>]
```



Conclusion:

Based on observation of all the models and their predictions, We should consider to go with Random Forest Classifier which has achieved maximum prediction abilities when compared to other models.

Model Accuracy Score = 0.99

f1 Score =0.879

AUC = 0.943

ap=0.931

In [185]:

```
df.to_csv("health care diabetes new.csv",index=False)
```