Predict Diabetes

import the necessary library and the data

In [56]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

diabetes=pd.read_csv('diabetes.csv')
print(diabetes.columns)
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [3]:

```
diabetes.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288

In [4]:

```
print("dimension  of the data: {}".format(diabetes.shape))
```

```
dimension  of the data: (768, 9)
```

In [5]:

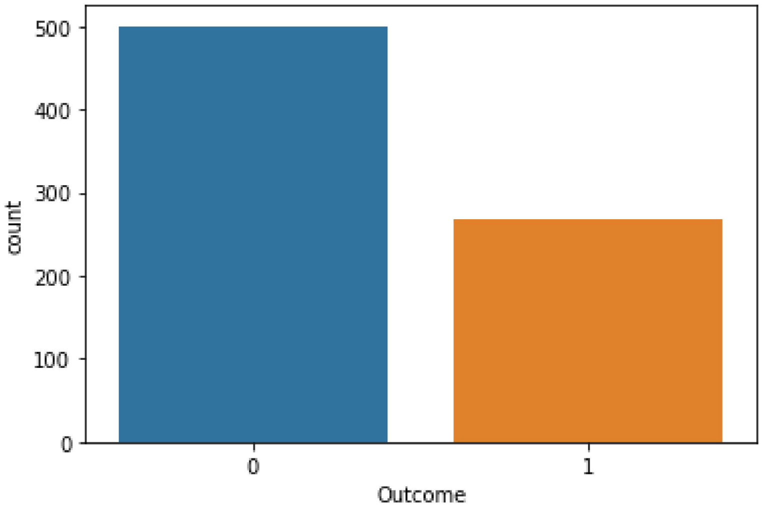
```
#grouping data based on outcome
print(diabetes.groupby('Outcome').size())
```

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

In [6]:
```
import seaborn as sns
sns.countplot(diabetes['Outcome'], label="Count")
```

```
C:\Users\Raj Krishna Mondal\anaconda3\lib\importlib\_bootstrap.py:219: Runtim
eWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expe
cted 192 from C header, got 216 from PyObject
  return f(*args, **kwds)
C:\Users\Raj Krishna Mondal\anaconda3\lib\importlib\_bootstrap.py:219: Runtim
eWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expe
cted 192 from C header, got 216 from PyObject
  return f(*args, **kwds)
C:\Users\Raj Krishna Mondal\anaconda3\lib\importlib\_bootstrap.py:219: Runtim
eWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expe
cted 192 from C header, got 216 from PyObject
  return f(*args, **kwds)
C:\Users\Raj Krishna Mondal\anaconda3\lib\importlib\_bootstrap.py:219: Runtim
eWarning: numpy.ufunc size changed, may indicate binary incompatibility. Expe
cted 192 from C header, got 216 from PyObject
  return f(*args, **kwds)
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x14107b34988>
```



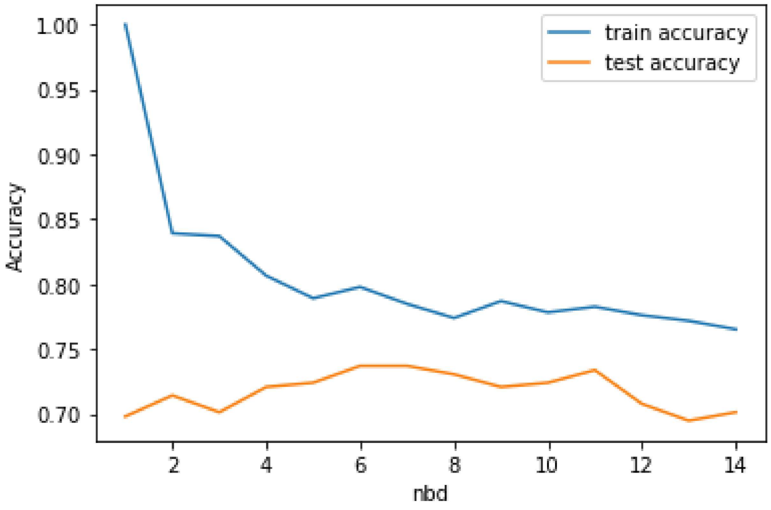
```
In [7]: # some information of our data
        diabetes.info()
```

```
<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
```

K-Nearest Neighbours to Predict Diabetes

In [82]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test=train_test_split(diabetes.loc[:,diabetes.colu

from sklearn.neighbors import KNeighborsClassifier

train_accuracy=[]
test_accuracy=[]

nbd=range(1,15)

for n_nbd in nbd:
    #build the model
    knn=KNeighborsClassifier(n_neighbors=n_nbd)
    knn.fit(x_train, y_train)

    #record the accuracy
    train_accuracy.append(knn.score(x_train, y_train))
    test_accuracy.append(knn.score(x_test, y_test))

plt.plot(nbd, train_accuracy, label="train accuracy")
plt.plot(nbd, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("nbd")

plt.legend()
```

Out[82]:
```
<matplotlib.legend.Legend at 0x14111920288>
```



In [75]:
```
knn=KNeighborsClassifier(n_neighbors=11)
knn.fit(x_train, y_train)
print(knn.score(x_train, y_train))
print(knn.score(x_test, y_test))
```

```
0.7866449511400652
0.7272727272727273
```

Decision Tree Classifier to Predict Diabetes

In [76]:

```
from sklearn.tree import DecisionTreeClassifier
tree=DecisionTreeClassifier(random_state=0)

tree.fit(x_train, y_train)
print(tree.score(x_train, y_train))
print(tree.score(x_test, y_test))
```

```
1.0
0.7077922077922078
```

In [77]:

```
tree=DecisionTreeClassifier(max_depth=3, random_state=0)
tree.fit(x_train, y_train)

print(tree.score(x_train, y_train))
print(tree.score(x_test, y_test))
```

```
0.7964169381107492
0.6883116883116883
```

Feature Importance in Decision Trees

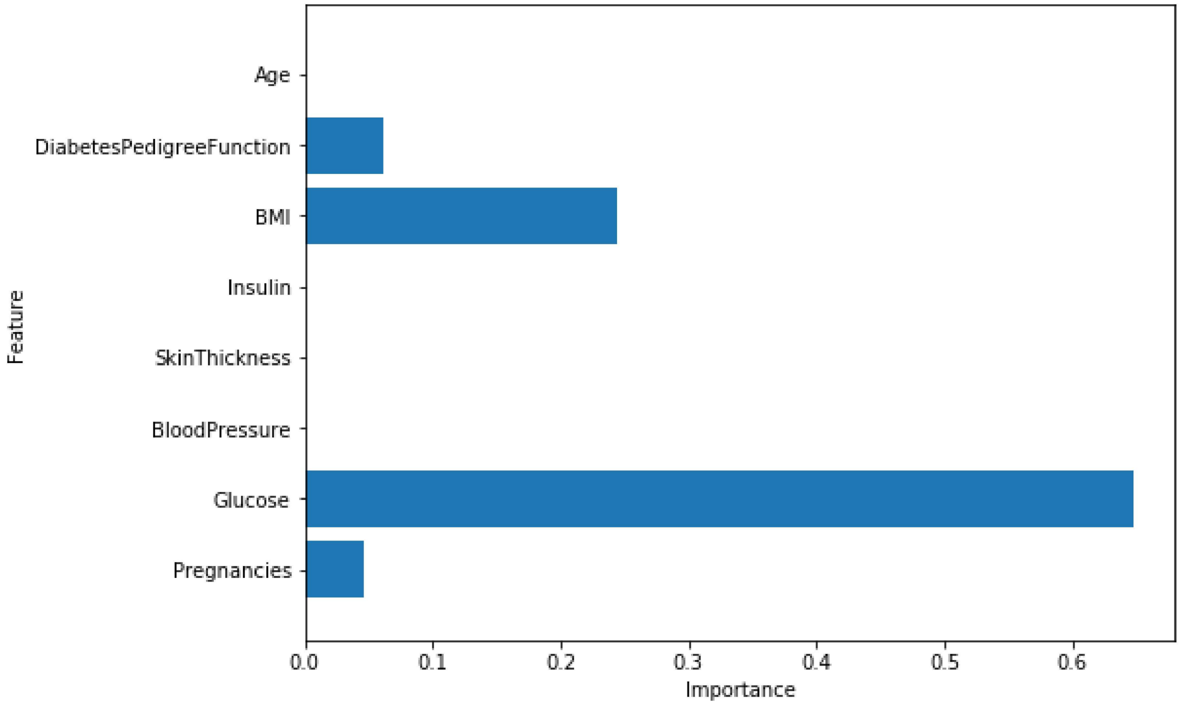
In [78]:

```
print(tree.feature_importances_)
```

```
[0.         0.74355317 0.         0.         0.         0.25644683
 0.         0.        ]
```

In [60]:

```
diabetes_features=diabetes.loc[:,diabetes.columns!='Outcome']
def plot_FI(model):
    plt.figure(figsize=(8,6))
    features=8
    plt.barh(range(features),model.feature_importances_)
    plt.ylabel("Feature")
    plt.xlabel("Importance")
    plt.yticks(np.arange(features), diabetes_features)

    plt.ylim(-1,features)

plot_FI(tree)
```



Deep Learning to Predict Diabetes

In [83]:

```
from sklearn.neural_network import MLPClassifier
mlp=MLPClassifier(random_state=45)

mlp.fit(x_train, y_train)

print(mlp.score(x_train, y_train))
print(mlp.score(x_test, y_test))
```

```
0.7717391304347826
0.7045454545454546
```

In [84]:

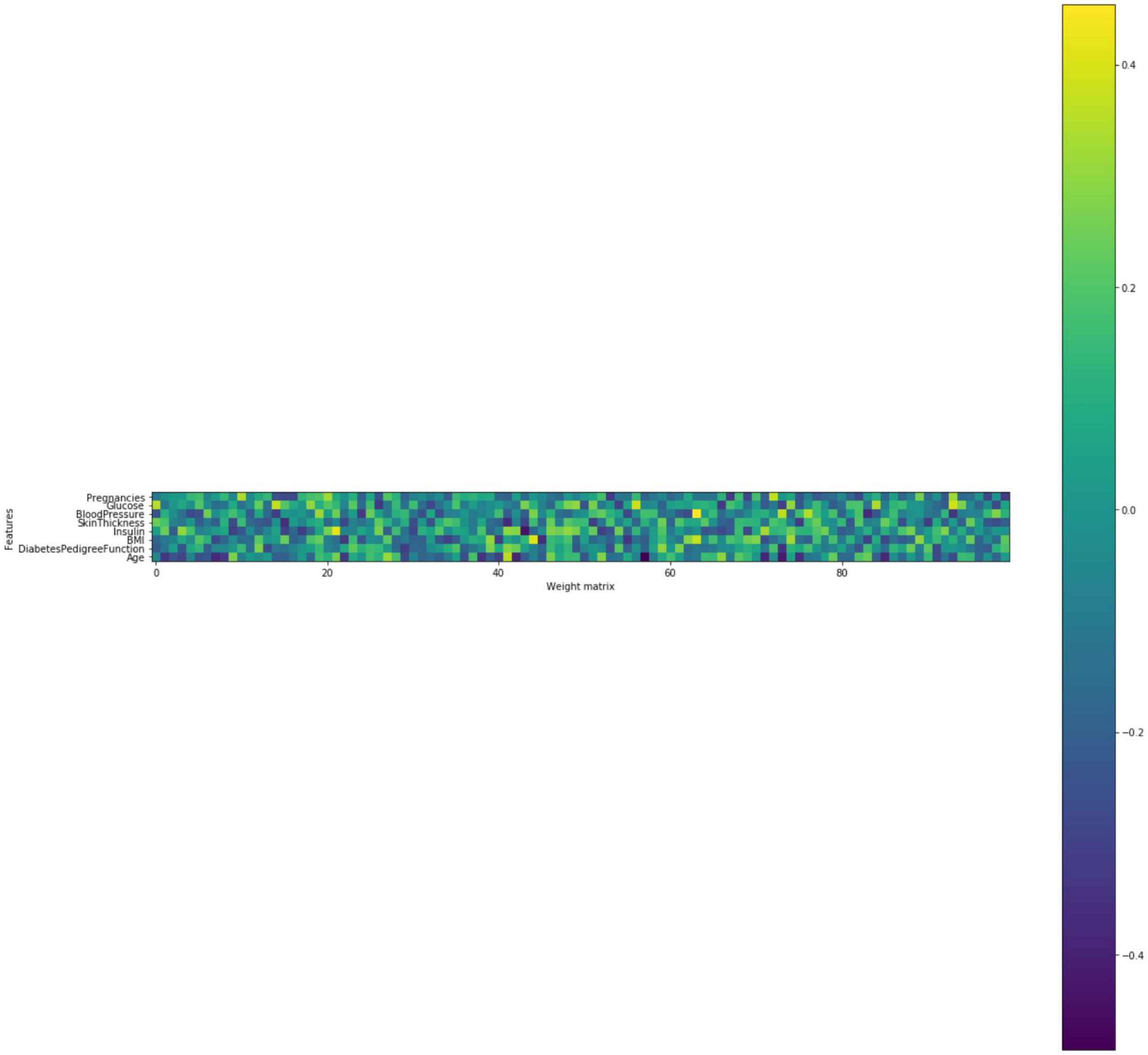
```
from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
x_train_scale=scale.fit_transform(x_train)
x_test_scale=scale.fit_transform(x_test)

mlp=MLPClassifier(random_state=0)

mlp.fit(x_train_scale, y_train)

print(mlp.score(x_train_scale, y_train))
print(mlp.score(x_test_scale, y_test))
```

```
0.8152173913043478
0.775974025974026
```

```
C:\Users\Raj Krishna Mondal\anaconda3\lib\site-packages\sklearn\neural_networ
k\_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Ma
ximum iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

In [87]:
```
plt.figure(figsize=(20,20))
plt.imshow(mlp.coefs_[0], interpolation='none', cmap='viridis')

plt.yticks(range(8),diabetes_features)
plt.xlabel("Weight matrix")
plt.ylabel("Features")

plt.colorbar()
```

Out[87]:
```
<matplotlib.colorbar.Colorbar at 0x1410b0d7d88>
```



In [ ]: