```
#Jayesh mali T084
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category = FutureWarning)
%matplotlib inline
diabetes = pd.read_csv('diabetes.csv')
print(diabetes.columns)
```

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

```
diabetes.head()
```

```
   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


   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

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

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

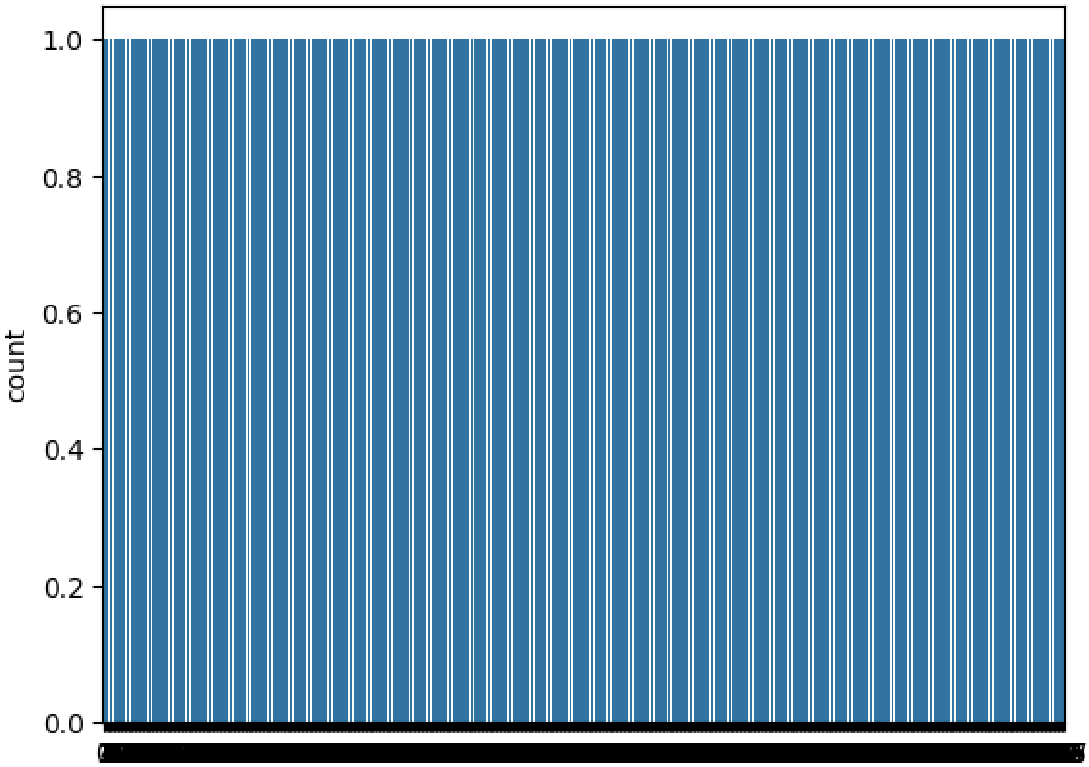
```
print(diabetes.groupby('Outcome').size())
```

```
Outcome
0    500
```

```
1    268
dtype: int64
```

```
sns.countplot(diabetes['Outcome'],label="Count")
```

```
<Axes: ylabel='count'>
```



```
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
```

Logistic Regression

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(diabetes.loc[:,
diabetes.columns != 'Outcome'], diabetes['Outcome'],
stratify=diabetes['Outcome'], random_state=66)
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(max_iter=3000).fit(X_train, y_train)

print("Training set score: {:.3f}".format(logreg.score(X_train,
y_train)))

print("Test set score: {:.3f}".format(logreg.score(X_test, y_test)))
```

```
Training set score: 0.785
Test set score: 0.771
```

```python
logreg001 = LogisticRegression(C=0.01,max_iter=3000).fit(X_train,
y_train)
print("Training set accuracy: {:.3f}".format(logreg001.score(X_train,
y_train)))
print("Test set accuracy: {:.3f}".format(logreg001.score(X_test,
y_test)))
```

```
Training set accuracy: 0.778
Test set accuracy: 0.755
```

```python
logreg100 = LogisticRegression(C=100,max_iter=3000).fit(X_train,
y_train)
print("Training set accuracy: {:.3f}".format(logreg100.score(X_train,
y_train)))
print("Test set accuracy: {:.3f}".format(logreg100.score(X_test,
y_test)))
```

```
Training set accuracy: 0.785
Test set accuracy: 0.766
```

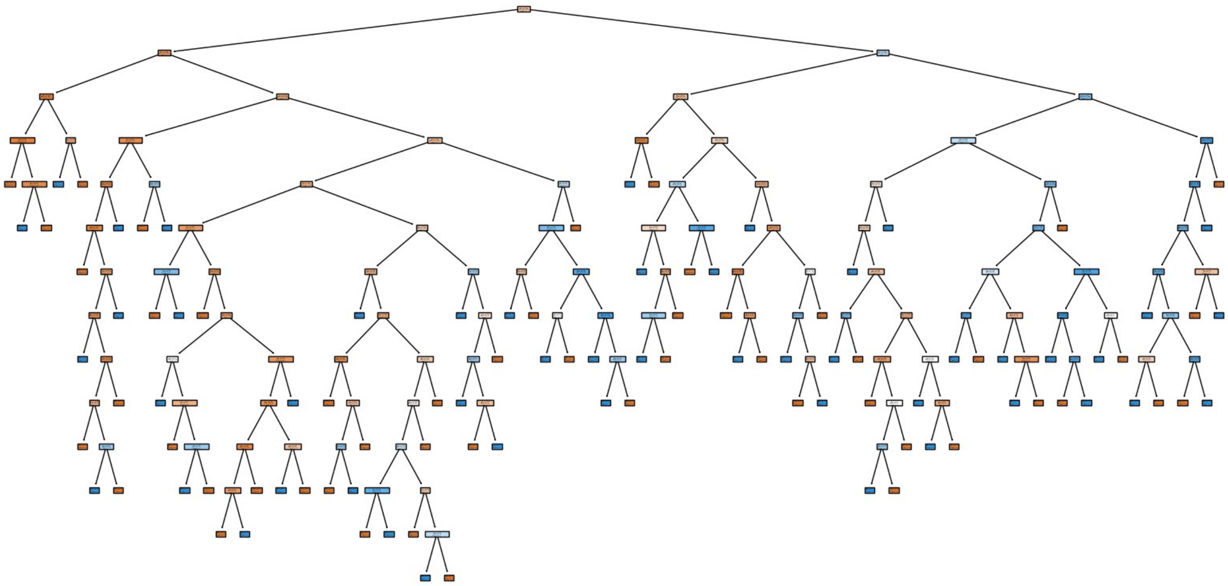
Decision Tree

```python
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(tree.score(X_train,
y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test,
y_test)))
```

```
Accuracy on training set: 1.000
Accuracy on test set: 0.714
```

```
from sklearn.tree import plot_tree

# Plot the decision tree
plt.figure(figsize=(20,10))
plot_tree(tree, filled=True, feature_names=X_train.columns,
class_names=['0', '1'])
plt.show()
```

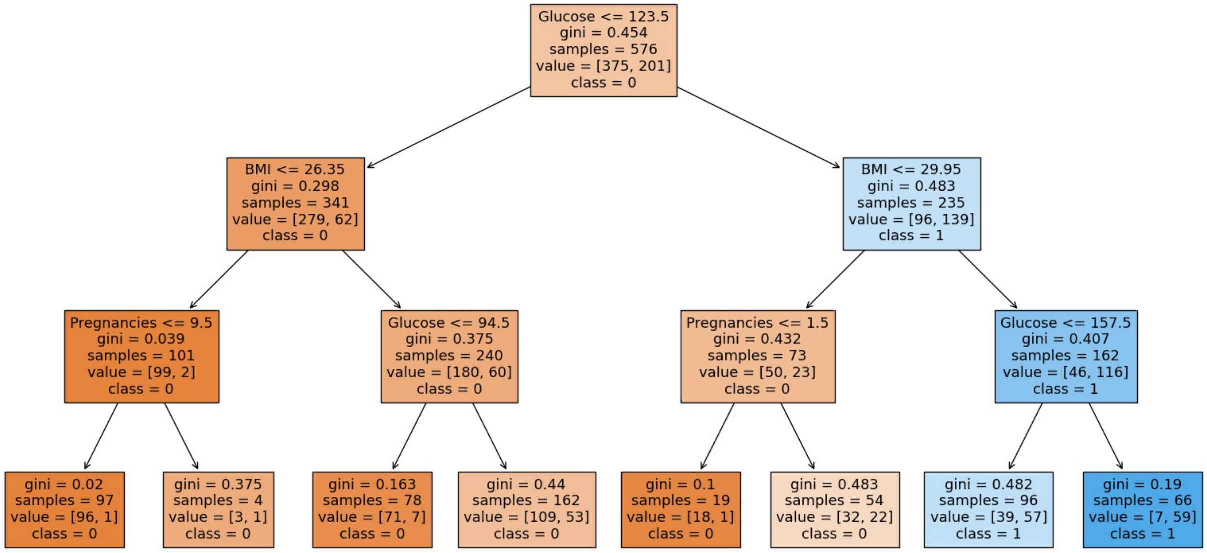


```
tree = DecisionTreeClassifier(max_depth=3, random_state=0)
tree.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(tree.score(X_train,
y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test,
y_test)))
```

```
Accuracy on training set: 0.773
Accuracy on test set: 0.740
```

```
from sklearn.tree import plot_tree

# Plot the decision tree
plt.figure(figsize=(20,10))
plot_tree(tree, filled=True, feature_names=X_train.columns,
class_names=['0', '1'])
plt.show()
```



```
print("Feature importances:\n{}".format(tree.feature_importances_))
```

```
Feature importances:
[0.04554275 0.6830362  0.         0.         0.         0.27142106
 0.         0.        ]
```