In [ ]:
```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
```

In [11]:
```
file_path = r"C:\Users\Yi Jun Zhuo\Downloads\diabetes.csv"
df = pd.read_csv(file_path)
print(df.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  
```

In [12]:
```
#replace the data is 0 to median
cols = ['Glucose', 'BloodPressure', 'SkinThickness','Insulin', 'BMI']
df[cols] = df[cols].replace(0, np.nan)
df.fillna(df.median(),inplace=True)
print("\nMissing values after cleaninng:\n", df.isnull(). sum())
```

```
Missing values after cleaninng:
 Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [13]:
```python
#create new column
df['BMI_Age'] = df['BMI']*df['Age']
df['Glucose_Insulin_ratio'] = df['Glucose']/(df['Insulin']+1)

print("\nNew features:\n",df[['BMI_Age', 'Glucose_Insulin_ratio']].head())
```

```
New features:
    BMI_Age  Glucose_Insulin_ratio
0   1680.0               1.174603
1    824.6               0.674603
2    745.6               1.452381
3    590.1               0.936842
4   1422.3               0.810651
```

In [ ]:
```python
#split pratice and test
X = df.drop('Outcome', axis =1)
Y = df['Outcome']
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=50, stratify=Y)
print("\nTraining size:", X_train.shape)
print("Test size:", X_test.shape)
```

```
Training size: (614, 10)
Test size: (154, 10)
```

In [28]:
```python
models = {
    "Decision Tree": DecisionTreeClassifier(max_depth=6, min_samples_split=50, class_weight='balanced', random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=300, max_depth=6, class_weight='balanced', random_state=42),
    "XGBoost": XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
```

```
}
results = {}
```

In [29]:

```
for name, model in models.items():
    model.fit(X_train, Y_train)
    Y_pred = model.predict(X_test)
    acc = accuracy_score(Y_test, Y_pred)
    roc = roc_auc_score(Y_test, Y_pred)
    print(f"\n{name} Performance:")
    print(f"Accuracy: {acc:.4f}")
    print(f"ROC AUC: {roc:.4f}")
    print("Classification Report:\n", classification_report(Y_test, Y_pred))
    print("Confusion Matrix:")
    print(pd.crosstab(Y_test, Y_pred, rownames=['Actual'], colnames=['Predicted']))
    results[name] = {'accuracy': acc, 'roc_auc': roc}
```

```
Decision Tree Performance:
Accuracy: 0.7468
ROC AUC: 0.7411
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.76      0.80       100
           1       0.62      0.72      0.67        54

    accuracy                           0.75       154
   macro avg       0.73      0.74      0.73       154
weighted avg       0.76      0.75      0.75       154

Confusion Matrix:
Predicted   0   1
Actual
0          76  24
1          15  39

Random Forest Performance:
Accuracy: 0.7338
ROC AUC: 0.7311
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.74      0.78       100
           1       0.60      0.72      0.66        54

    accuracy                           0.73       154
   macro avg       0.72      0.73      0.72       154
weighted avg       0.75      0.73      0.74       154

Confusion Matrix:
Predicted   0   1
Actual
0          74  26
1          15  39

XGBoost Performance:
Accuracy: 0.7532
ROC AUC: 0.7206
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.83      0.81       100
           1       0.66      0.61      0.63        54

    accuracy                           0.75       154
   macro avg       0.73      0.72      0.72       154
weighted avg       0.75      0.75      0.75       154

Confusion Matrix:
Predicted   0   1
Actual
0          83  17
1          21  33
```

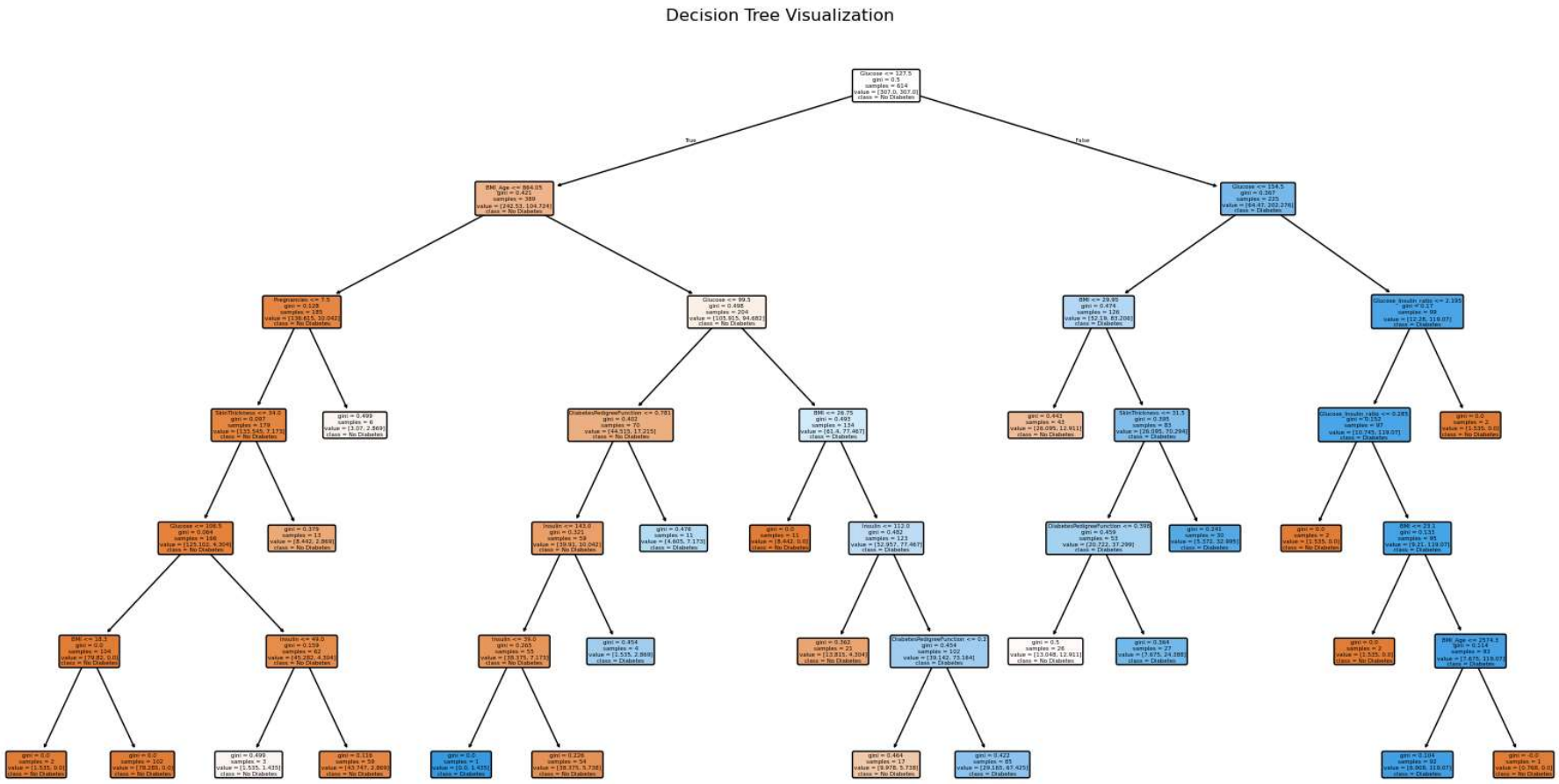
```
c:\Users\Yi Jun Zhuo\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [12:01:11] WARNING: C:\actions-runner\_w
ork\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

In [30]:
```
acc_df = pd.DataFrame(results).T[['accuracy', 'roc_auc']].sort_values(by='accuracy', ascending=False)
print("\nModel Comparison:\n", acc_df)
```

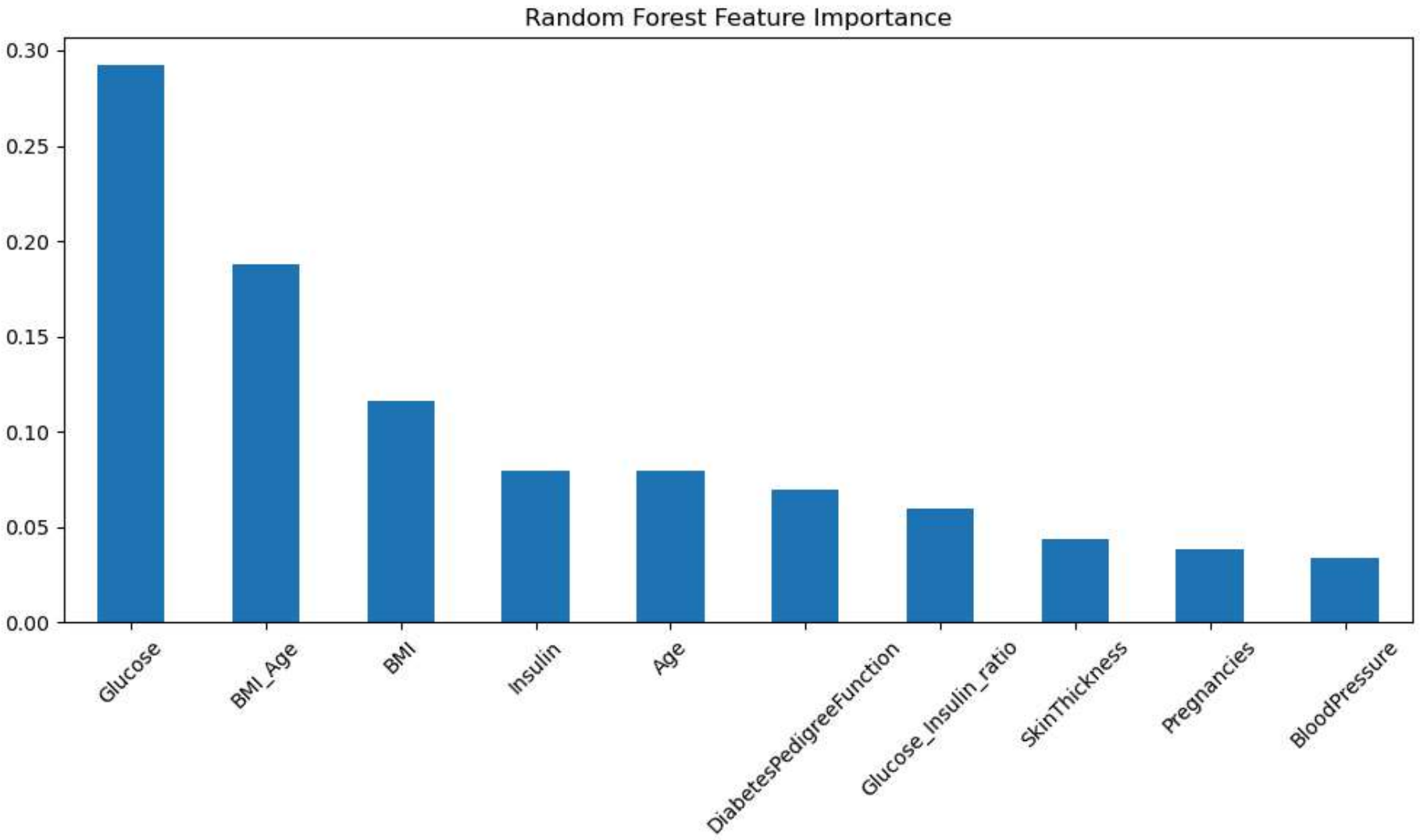
```
Model Comparison:
                accuracy   roc_auc
XGBoost         0.753247  0.720556
Decision Tree   0.746753  0.741111
Random Forest   0.733766  0.731111
```

In [31]:
```
plt.figure(figsize=(20,10))
plot_tree(models['Decision Tree'], feature_names = X.columns, class_names = ['No Diabetes','Diabetes'], filled = True, rounded
plt.title("Decision Tree Visualization")
```

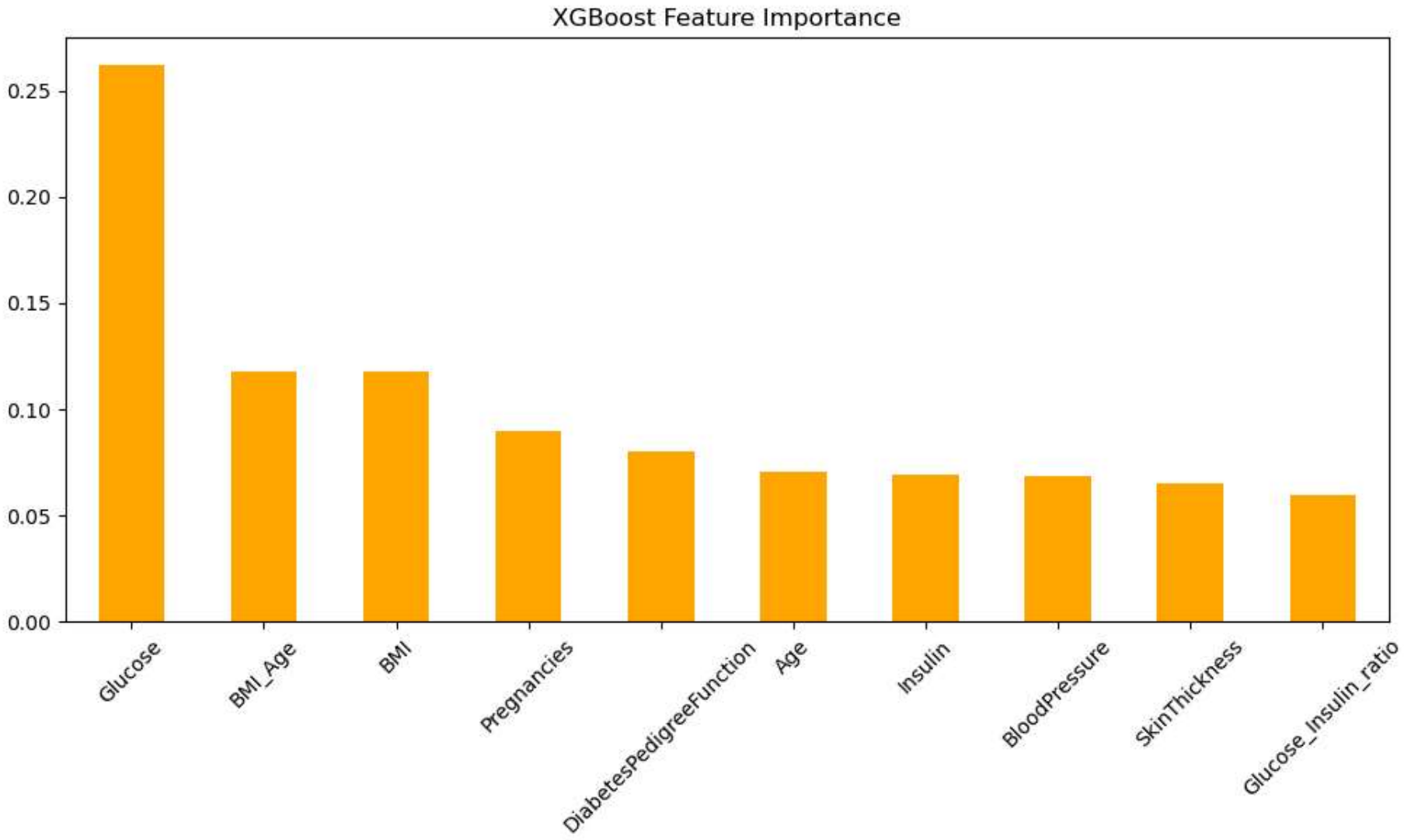
Out[31]:
```
Text(0.5, 1.0, 'Decision Tree Visualization')
```



In [33]:
```
importances_rf = pd.Series(models['Random Forest'].feature_importances_, index=X.columns).sort_values(ascending=False)
plt.figure(figsize=(10,6))
importances_rf.plot(kind='bar')
plt.title("Random Forest Feature Importance")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [34]:
```
importances_xgb = pd.Series(models['XGBoost'].feature_importances_, index=X.columns).sort_values(ascending=False)
plt.figure(figsize=(10,6))
importances_xgb.plot(kind='bar', color='orange')
plt.title("XGBoost Feature Importance")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


XGBoost Feature Importance
0.25
0.20
0.15
0.10
0.05
0.00
Glucose
BMI_Age
BMI
Pregnancies
DiabetesPedigreeFunction
Age
Insulin
BloodPressure
SkinThickness
Glucose_Insulin_ratio