In [3]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, accuracy_score, precision_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.tree import plot_tree
from sklearn.preprocessing import LabelEncoder
import warnings

warnings.filterwarnings('ignore')
```

In [6]:
```
# load and read the dataset into a DataFrame
df = pd.read_csv('D:\ericka may coronel\SIMULATORS\ANACONDA\consolidated_philip

# Display the first few ros of the dataset
df.head()
```

Out[6]:

	regDesc	agr_wage_farm_workers_allgender_2015	agr_wage_farm_workers_male_2015	agr_wa
0	Armm	162.89	163.65	
1	Bicol Region	167.99	169.95	
2	Cagayan Valley	228.77	232.64	
3	Calabarzon	230.92	231.45	
4	Car	206.68	211.04	

In [7]:
```
# Check for missing values
df.isnull().sum()

# Handle missing values (if any)
df.dropna(inplace=True)

# Check for duplicate rows
df.duplicated().sum()

# Drop duplicates if any
df = df.drop_duplicates()
```

In [105]:
```
df
```

Out[105]:

	regDesc	agr_wage_farm_workers_allgender_2015	agr_wage_farm_workers_male_2015	agr
0	Armm	162.89	163.65	
1	Bicol Region	167.99	169.95	
2	Cagayan Valley	228.77	232.64	
4	Car	206.68	211.04	
5	Caraga	194.46	195.44	
6	Central Luzon	257.97	259.04	
7	Central Visayas	156.17	160.65	
8	Davao Region	168.68	169.83	
9	Eastern Visayas	157.49	159.25	
10	Ilocos Region	237.26	239.19	
12	Northern Mindanao	159.12	160.07	
13	Soccsksargen	164.77	166.75	
14	Western Visayas	165.28	167.77	
15	Zamboanga Peninsula	157.37	158.55	

In [8]:
```
# Perfrom descriptive statistics
print("\nDescriptive Statistics of the Dataset: ")
df.describe()
```

```
Descriptive Statistics of the Dataset:
```

Out[8]:

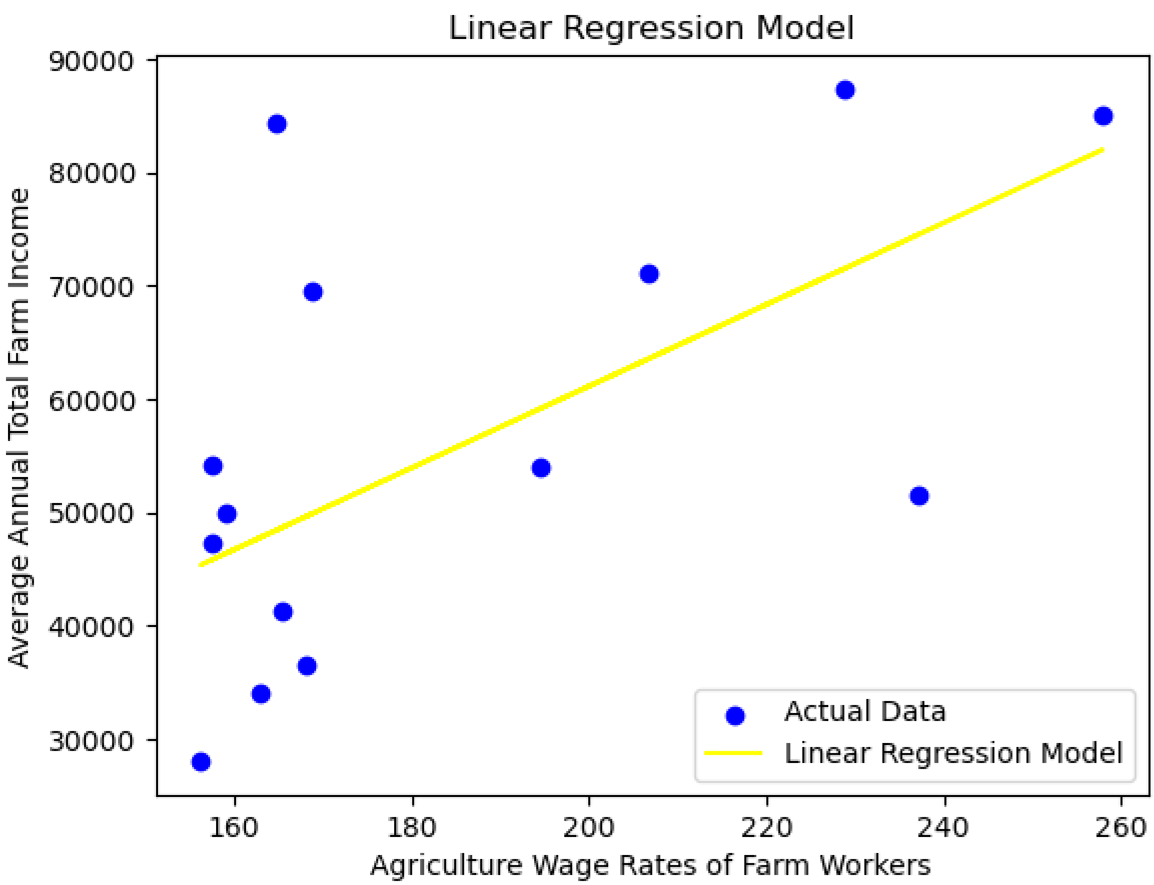
	agr_wage_farm_workers_allgender_2015	agr_wage_farm_workers_male_2015	agr_wage_farm
count	14.000000	14.000000	
mean	184.635714	186.701429	
std	34.410445	34.589990	
min	156.170000	158.550000	
25%	160.062500	161.400000	
50%	166.635000	168.800000	
75%	203.625000	207.140000	
max	257.970000	259.040000	

1. Linear Regression Model

In [9]:
```
#Implementing linear regression model to analyze relationships between variable

# Define variables
var1 = 'agr_wage_farm_workers_allgender_2015'
var2 = 'avg_annual_farm_incm_farm_households_02_03'

# Implement linear regression to selected features
X = df[[var1]]
y = df[[var2]]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

# Create and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model using mean squared error
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 138225856.43588775
```

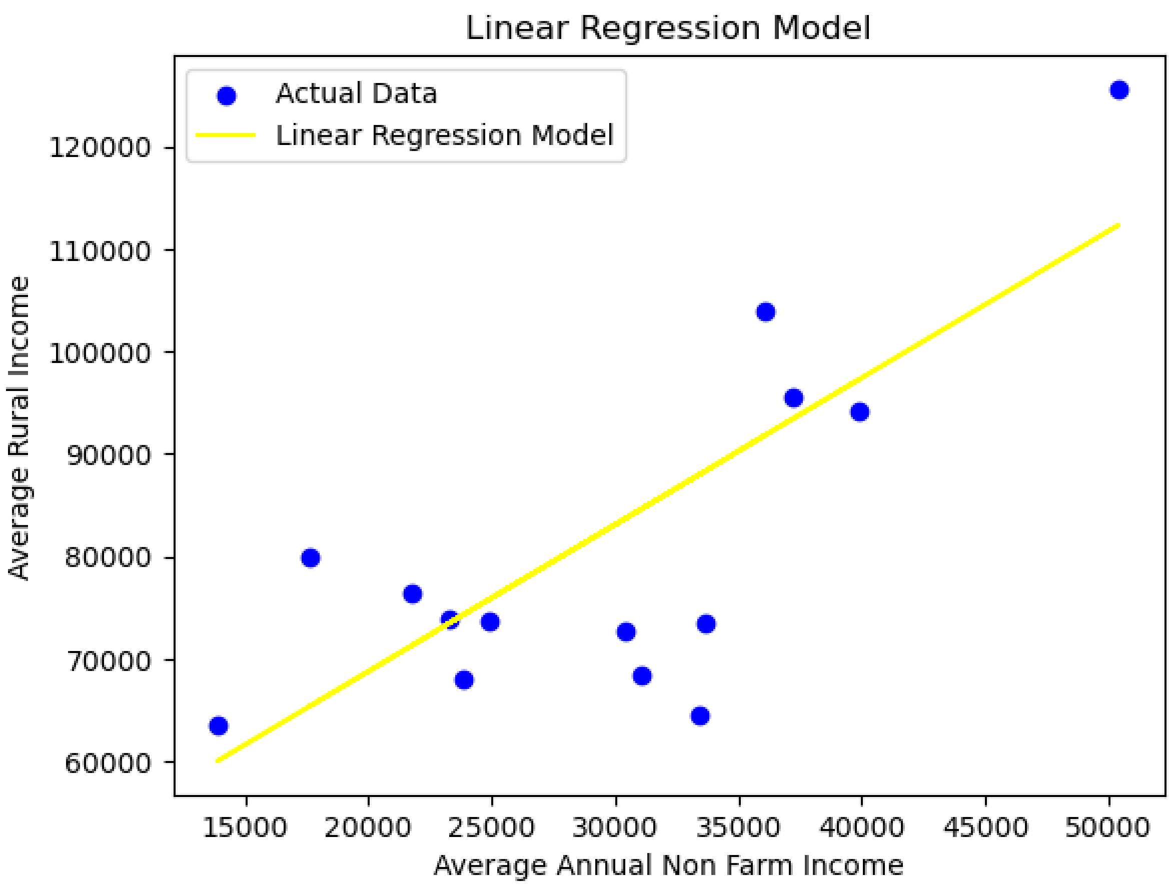
In [10]:
```
# Visualize the relation between variable1 and variable2
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, model.predict(X), color='yellow', label='Linear Regression Model')
plt.title('Linear Regression Model')
plt.xlabel('Agriculture Wage Rates of Farm Workers')
plt.ylabel('Average Annual Total Farm Income')
plt.legend()
plt.show()
```



In [109]:
```
#Implementing linear regression model to analyze relationships between variable

# Define variables
var3 = 'avg_annual_non_farm_incm_farm_households_02_03'
var4 = 'avg_rural_income_2000'

# Implement linear regression to selected features
X = df[[var3]]
y = df[[var4]]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

# Create and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model using mean squared error
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 114552777.79321332
```

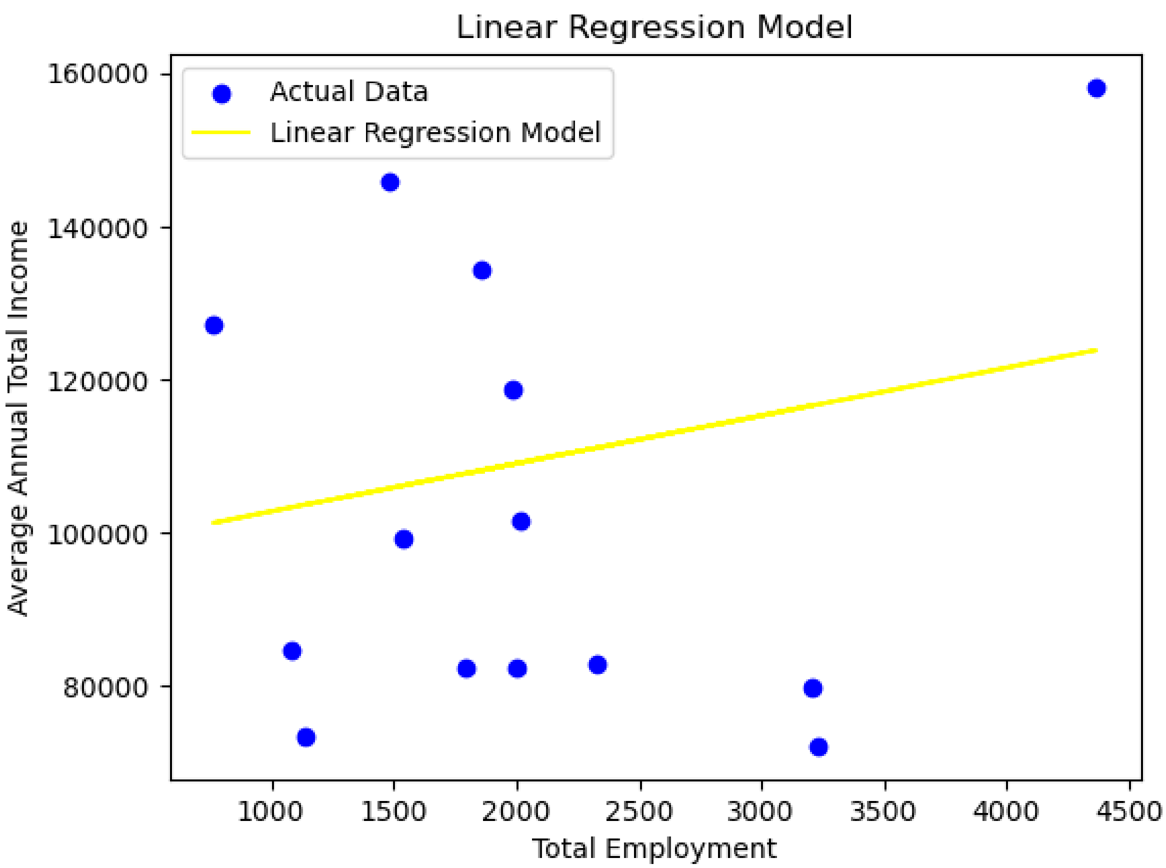
In [110]:
```
# Visualize the relation between variable3 and variable4
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, model.predict(X), color='yellow', label='Linear Regression Model')
plt.title('Linear Regression Model')
plt.xlabel('Average Annual Non Farm Income')
plt.ylabel('Average Rural Income')
plt.legend()
plt.show()
```



In [111]:
```
#Implementing linear regression model to analyze relationships between variable

# Define variables
var5 = 'total_emply_2016'
var6 = 'avg_annual_total_incm_farm_households_02_03'

# Implement linear regression to selected features
X = df[[var5]]
y = df[[var6]]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

# Create and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model using mean squared error
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 991171396.6819845
```

In [112]:
```
# Visualize the relation between variable5 and variable6
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, model.predict(X), color='yellow', label='Linear Regression Model')
plt.title('Linear Regression Model')
plt.xlabel('Total Employment')
plt.ylabel('Average Annual Total Income')
plt.legend()
plt.show()
```

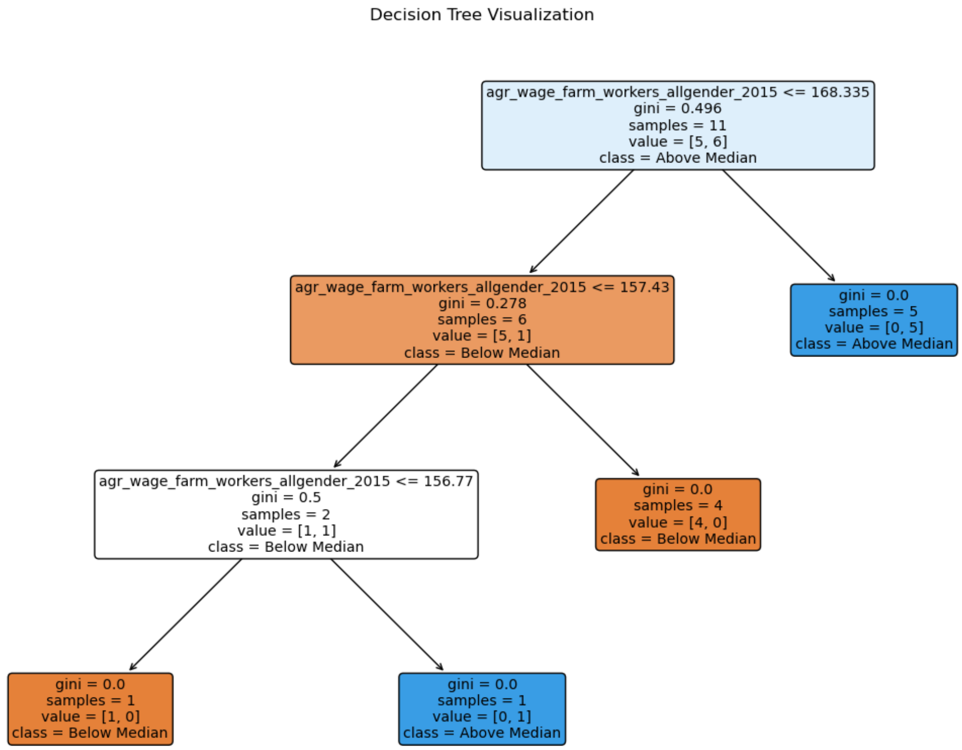


2. Classification Model (Decision Tree Model)

In [20]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_s

# Define variables
var1 = 'agr_wage_farm_workers_allgender_2015'
var2 = 'avg_annual_farm_incm_farm_households_02_03'

# Implement Decision Tree Classification model to selected features
X = df[[var1]]
y = (df[[var2]] > df[[var2]].median()).astype(int) # Convert to binary based or

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randon

# Create and train the Decision Tree model
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

# Predictions on the test set
y_pred = dt_model.predict(X_test)

# Evaluate the Decision Tree model using accuracy, precision, recall, and F1 sc
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

```
Accuracy: 0.3333333333333333
Precision: 0.0
Recall: 0.0
F1 Score: 0.0
```

In [21]:
```
# Visualize the Decision Tree
plt.figure(figsize=(15, 10))
plot_tree(dt_model, feature_names=X.columns.tolist(), class_names=['Below Media
plt.title('Decision Tree Visualization')
plt.show()
```

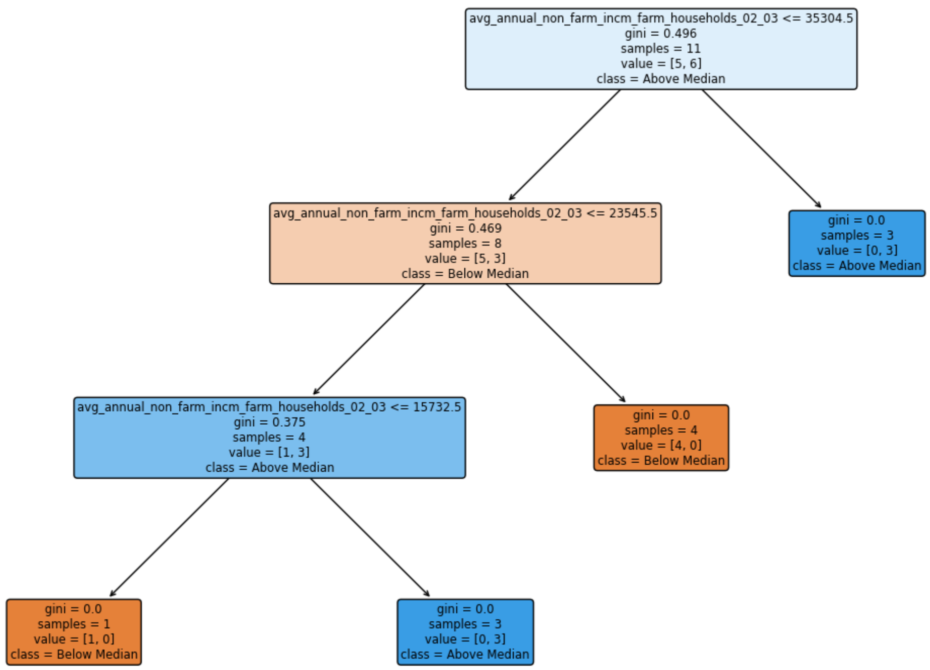


In [22]:
```
#Implementing Decision Tree Classification model to analyze relationships betwe

# Define variables
var3 = 'avg_annual_non_farm_incm_farm_households_02_03'
var4 = 'avg_rural_income_2000'

# Implement Decision Tree Classification model to selected features
X = df[[var3]]
y = (df[[var4]] > df[[var4]].median()).astype(int) # Convert to binary based or

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randon

# Create and train the Decision Tree model
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

# Predictions on the test set
y_pred = dt_model.predict(X_test)

# Evaluate the Decision Tree model using accuracy, precision, recall, and F1 sc
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

In [23]:
```
# Visualize the Decision Tree
plt.figure(figsize=(15, 10))
plot_tree(dt_model, feature_names=X.columns.tolist(), class_names=['Below Media
plt.title('Decision Tree Visualization')
plt.show()
```

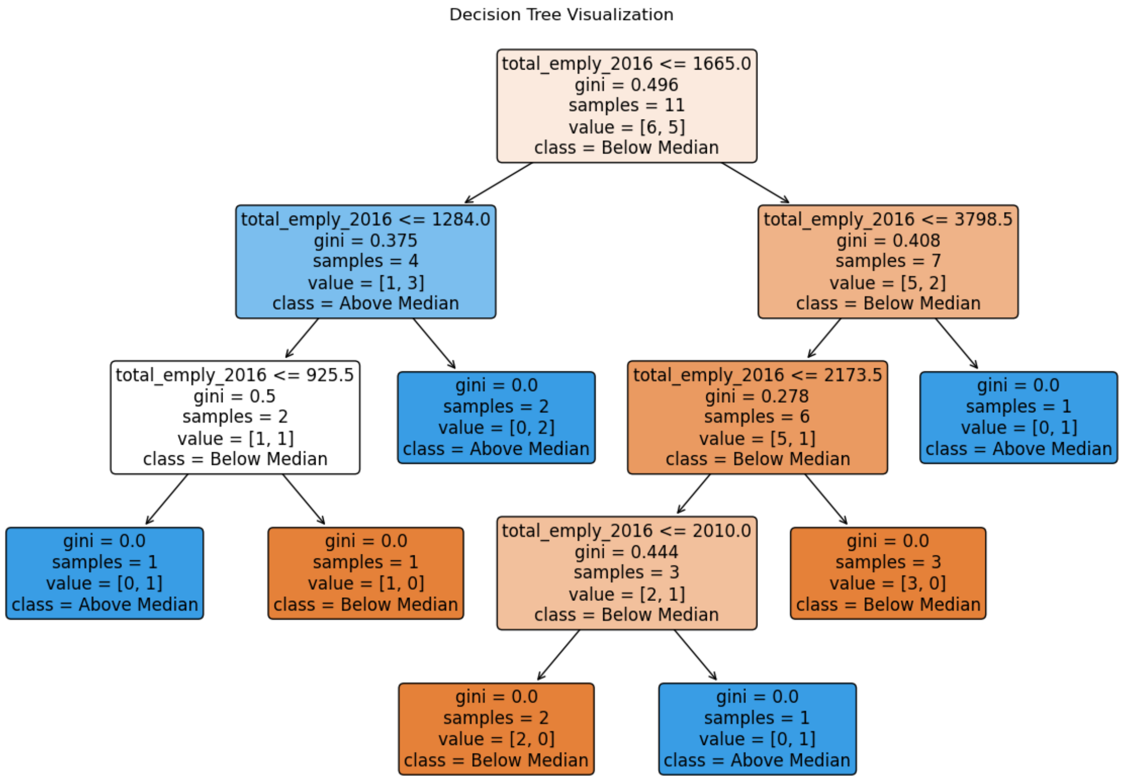
Decision Tree Visualization



In [16]:
```
#Implementing Decision Tree Classification model to analyze relationships betwe

# Define variables
var5 = 'total_emply_2016'
var6 = 'avg_annual_total_incm_farm_households_02_03'

# Implement Decision Tree Classification model to selected features
X = df[[var5]]
y = (df[[var6]] > df[[var6]].median()).astype(int) # Convert to binary based or

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randor

# Create and train the Decision Tree model
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

# Predictions on the test set
y_pred = dt_model.predict(X_test)

# Evaluate the Decision Tree model using accuracy, precision, recall, and F1 sc
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

```
Accuracy: 0.3333333333333333
Precision: 0.0
Recall: 0.0
F1 Score: 0.0
```

In [17]:
```
# Visualize the Decision Tree
plt.figure(figsize=(15, 10))
plot_tree(dt_model, feature_names=X.columns.tolist(), class_names=['Below Media
plt.title('Decision Tree Visualization')
plt.show()
```



In [ ]:

In [ ]: