In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_r
from sklearn.preprocessing import label_binarize

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

In [2]:
```
df = pd.read_csv("diabetes_012_health_indicators_BRFSS2015.csv")
df.head()
```

Out[2]:

	Diabetes_012	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDise
0	0.0	1.0	1.0	1.0	40.0	1.0	0.0	
1	0.0	0.0	0.0	0.0	25.0	1.0	0.0	
2	0.0	1.0	1.0	1.0	28.0	0.0	0.0	
3	0.0	1.0	0.0	1.0	27.0	0.0	0.0	
4	0.0	1.0	1.0	1.0	24.0	0.0	0.0	

5 rows × 22 columns

In [3]:
```
X = df.drop("Diabetes_012", axis=1)
y = df["Diabetes_012"]
```

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

In [6]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Logistic Regression

In [7]:
```
lr = LogisticRegression(max_iter=2000)
lr.fit(X_train_scaled, y_train)
pred_lr = lr.predict(X_test_scaled)
```

KNN

In [27]:
```
knn = KNeighborsClassifier(n_neighbors=7,weights='uniform')
knn.fit(X_train_scaled, y_train)
pred_knn = knn.predict(X_test_scaled)
```

RandomForestClassifier

In [29]:
```
rf = RandomForestClassifier(n_estimators=100, max_depth=8, random_state=42)
rf.fit(X_train, y_train)
pred_rf = rf.predict(X_test)
```

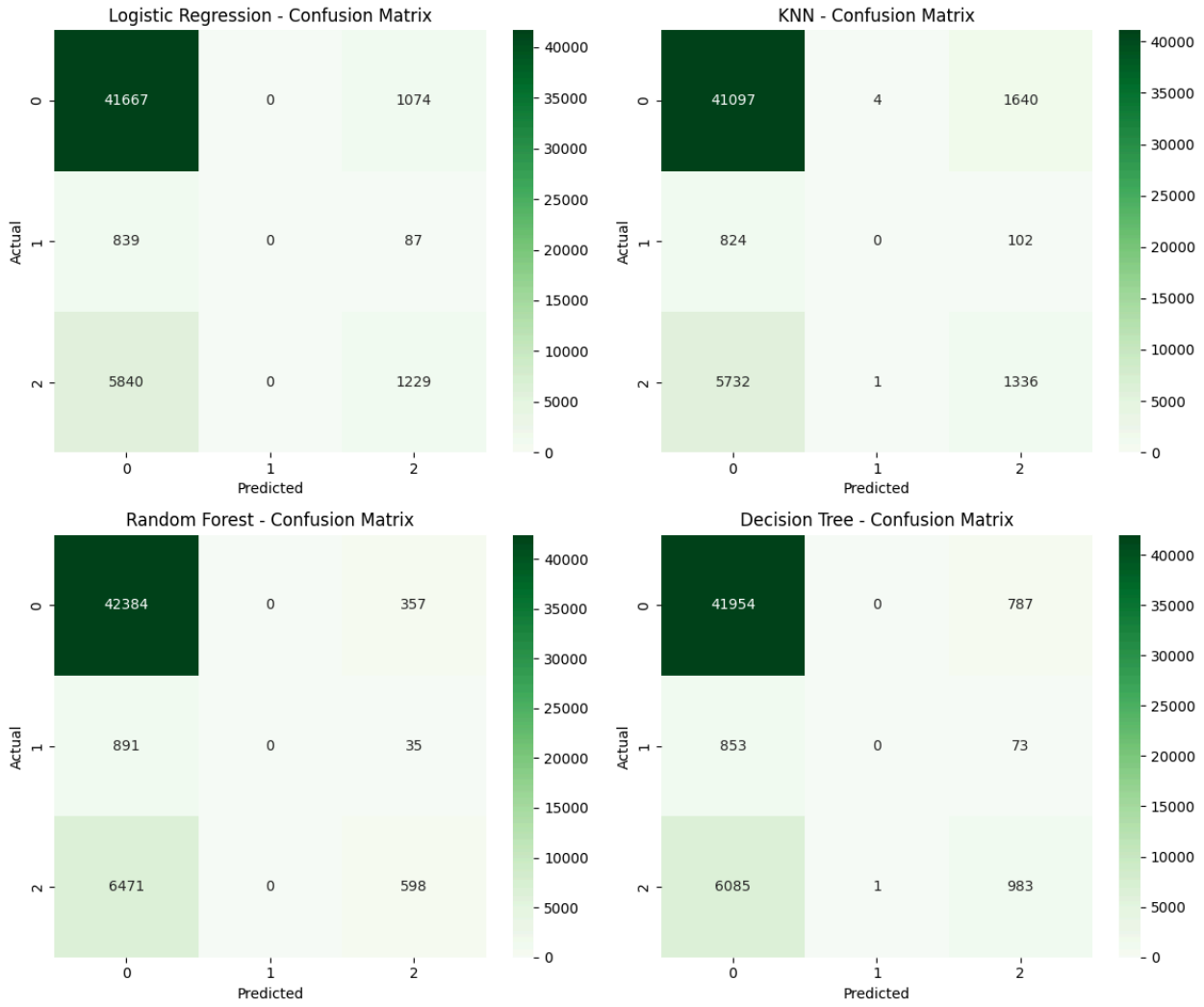
Decision Tree

In [10]:
```
dt = DecisionTreeClassifier(max_depth=8, random_state=42)
dt.fit(X_train, y_train)
pred_dt = dt.predict(X_test)
```

In [11]:
```
print("Logistic Regression Accuracy:", accuracy_score(y_test, pred_lr))
print("KNN Accuracy:", accuracy_score(y_test, pred_knn))
print("Random Forest Accuracy:", accuracy_score(y_test, pred_rf))
print("Decision Tree Accuracy:", accuracy_score(y_test, pred_dt))
```

```
Logistic Regression Accuracy: 0.8454746136865342
KNN Accuracy: 0.8363489435509303
Random Forest Accuracy: 0.8471696625670135
Decision Tree Accuracy: 0.8462827183853674
```

In [12]:
```
models = {
    "Logistic Regression": pred_lr,
    "KNN": pred_knn,
    "Random Forest": pred_rf,
    "Decision Tree": pred_dt
}
plt.figure(figsize=(12, 10))
for i, (name, preds) in enumerate(models.items(), 1):
    plt.subplot(2, 2, i)
    cm = confusion_matrix(y_test, preds)
    sns.heatmap(cm, annot=True, fmt="d", cmap="Greens")
    plt.title(f"{name} - Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```



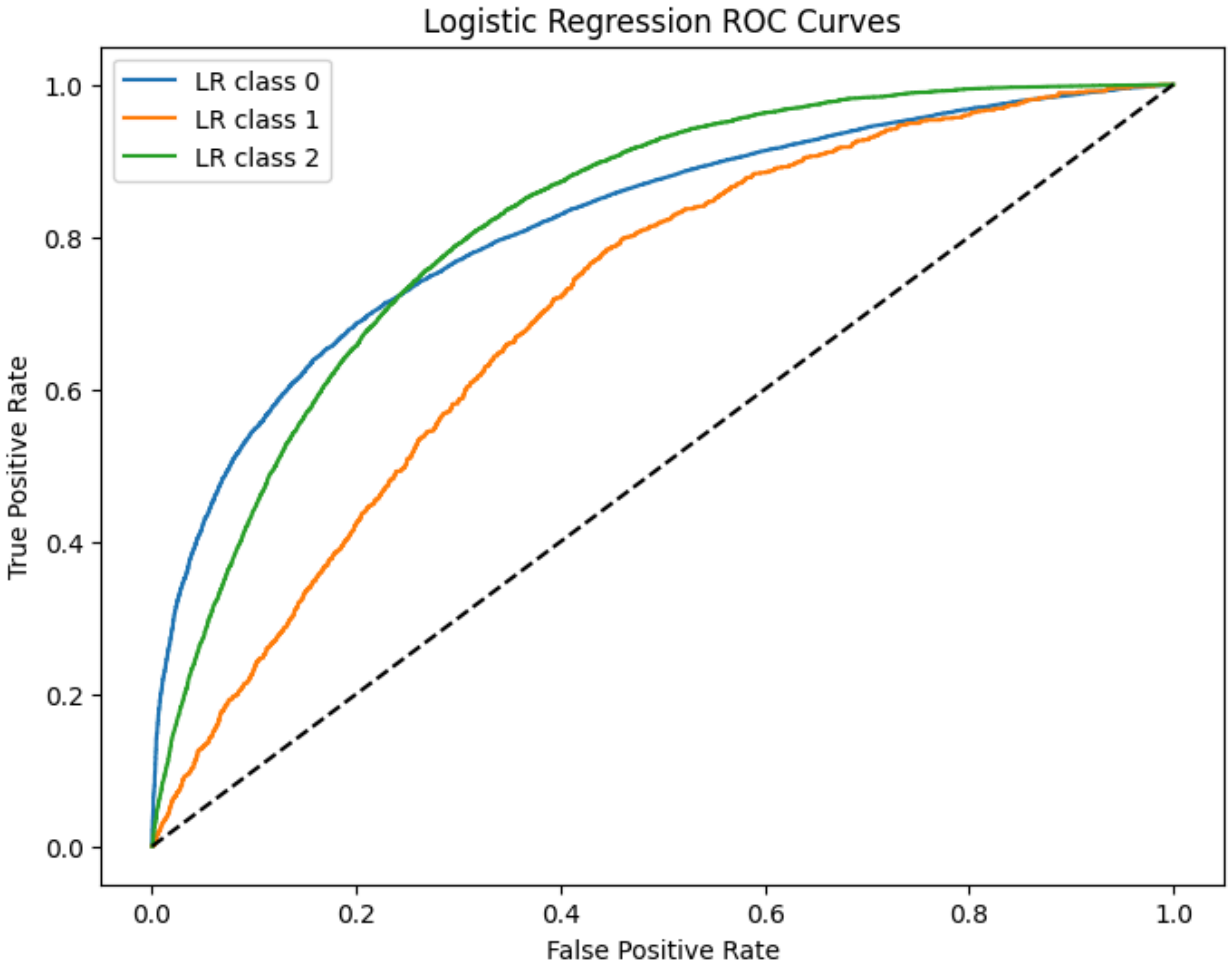
In [13]:
```
classes = [0, 1, 2]
y_test_bin = label_binarize(y_test, classes=classes)

plt.figure(figsize=(10, 7))
```

Out[13]:
```
<Figure size 1000x700 with 0 Axes>
```

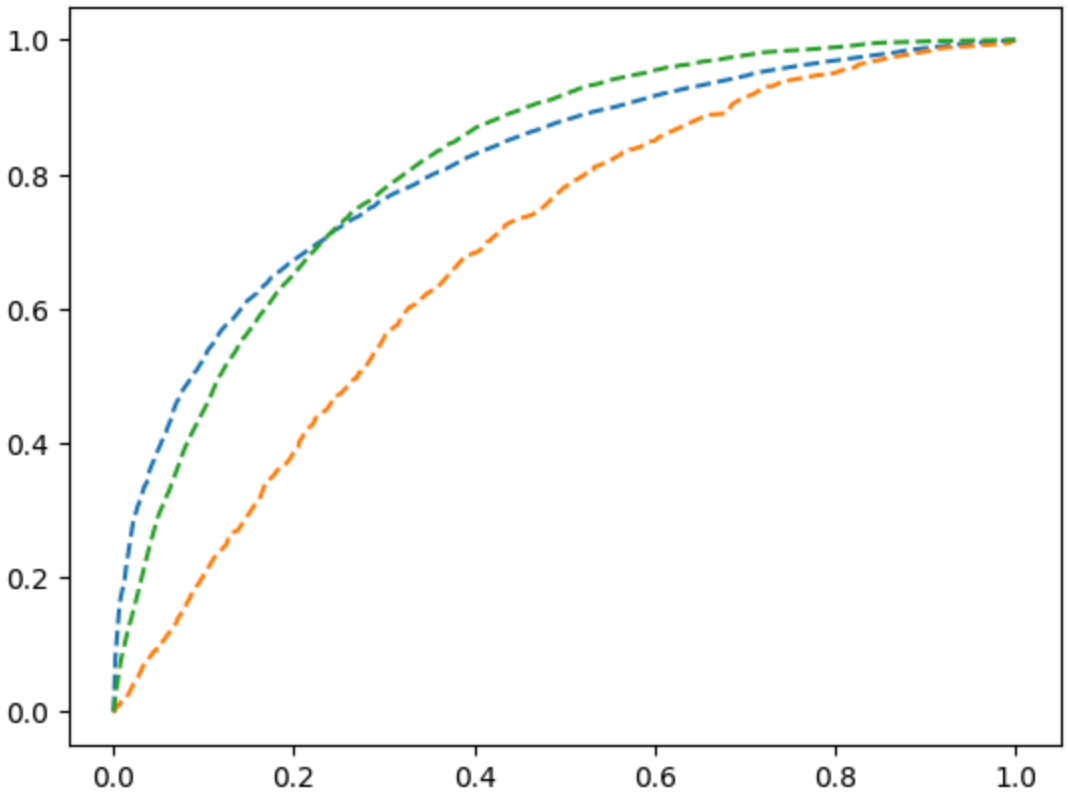
<Figure size 1000x700 with 0 Axes>

In [15]:
```
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt

# Binarize the test labels for multiclass ROC
classes = [0, 1, 2]
y_test_bin = label_binarize(y_test, classes=classes)

# Predict probabilities
probs_lr = lr.predict_proba(X_test_scaled)

# Plot ROC curves
plt.figure(figsize=(8,6))
```

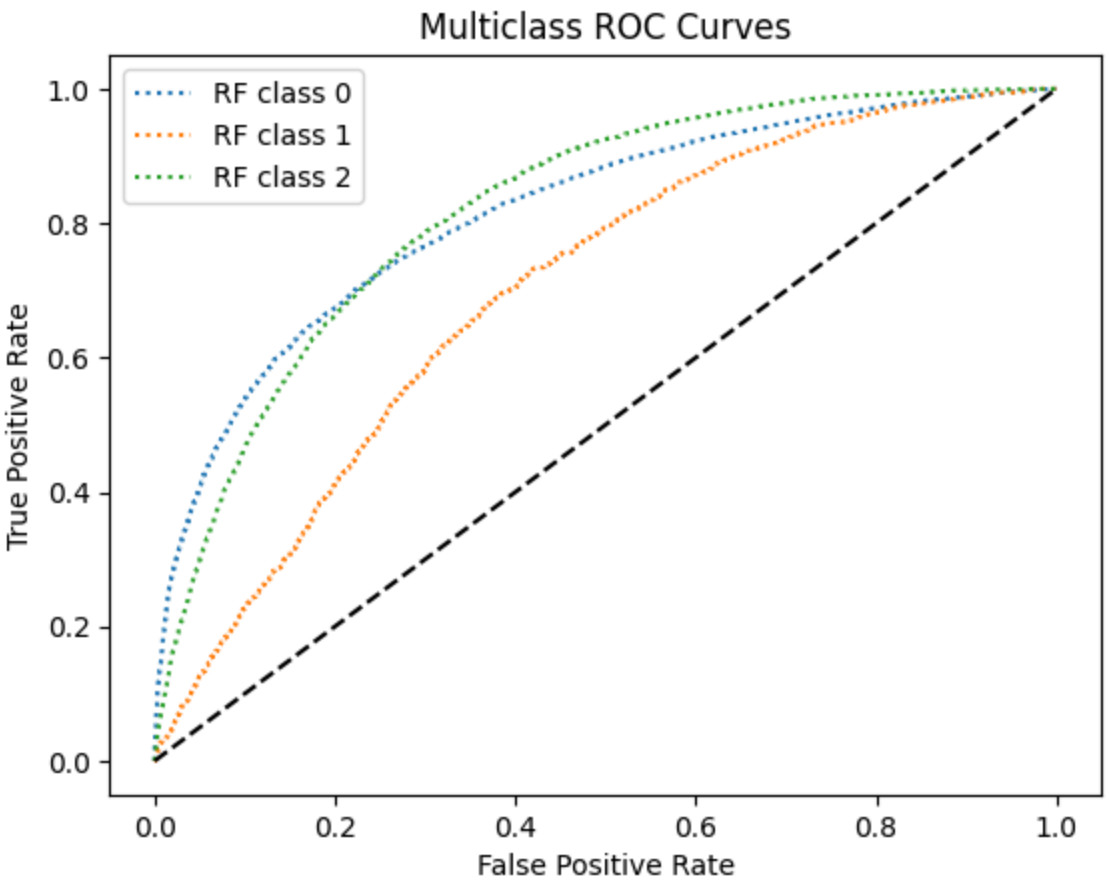
```
for i in range(3):
    fpr, tpr, _ = roc_curve(y_test_bin[:, i], probs_lr[:, i])
    plt.plot(fpr, tpr, label=f"LR class {i}")

plt.plot([0,1], [0,1], 'k--')  # diagonal line
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Logistic Regression ROC Curves")
plt.legend()
plt.show()
```



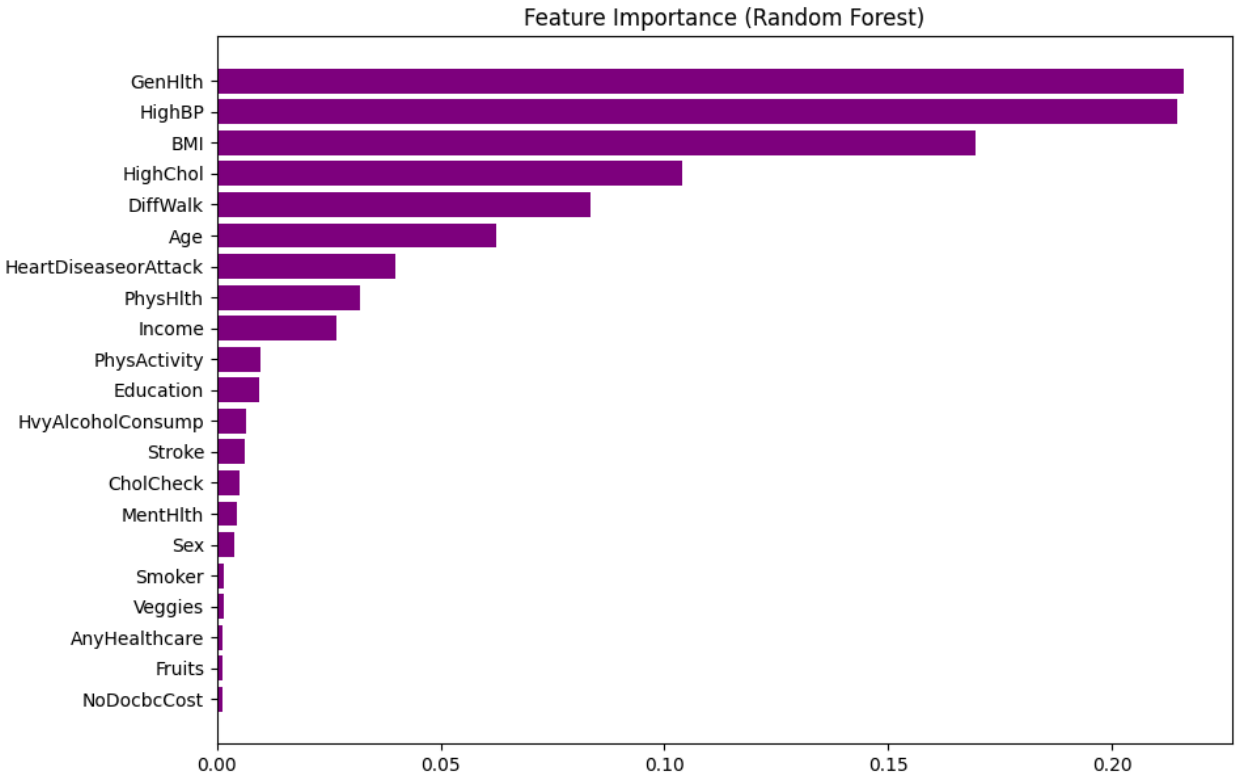
In [18]:
```
probs_dt = dt.predict_proba(X_test)
for i in range(3):
    fpr, tpr, _ = roc_curve(y_test_bin[:, i], probs_dt[:, i])
    plt.plot(fpr, tpr, '--', label=f"DT class {i}")
```



In [19]:
```
probs_rf = rf.predict_proba(X_test)
for i in range(3):
    fpr, tpr, _ = roc_curve(y_test_bin[:, i], probs_rf[:, i])
    plt.plot(fpr, tpr, ':', label=f"RF class {i}")

plt.plot([0, 1], [0, 1], 'k--')
plt.title("Multiclass ROC Curves")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```

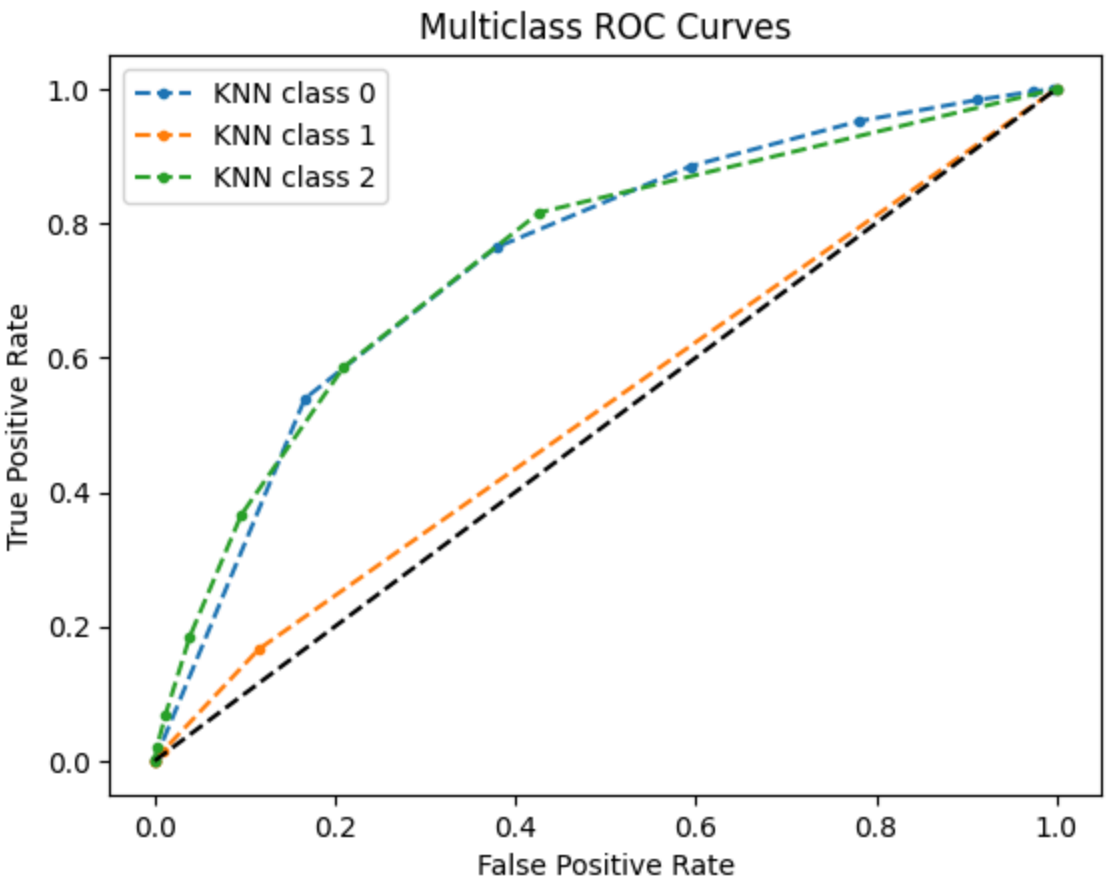


In [32]:
```
plt.figure(figsize=(10, 7))
importances = rf.feature_importances_
indices = np.argsort(importances)[::-1]
plt.barh(X.columns[indices], importances[indices], color='purple')
plt.title("Feature Importance (Random Forest)")
plt.gca().invert_yaxis()
plt.show()
```



In [28]:

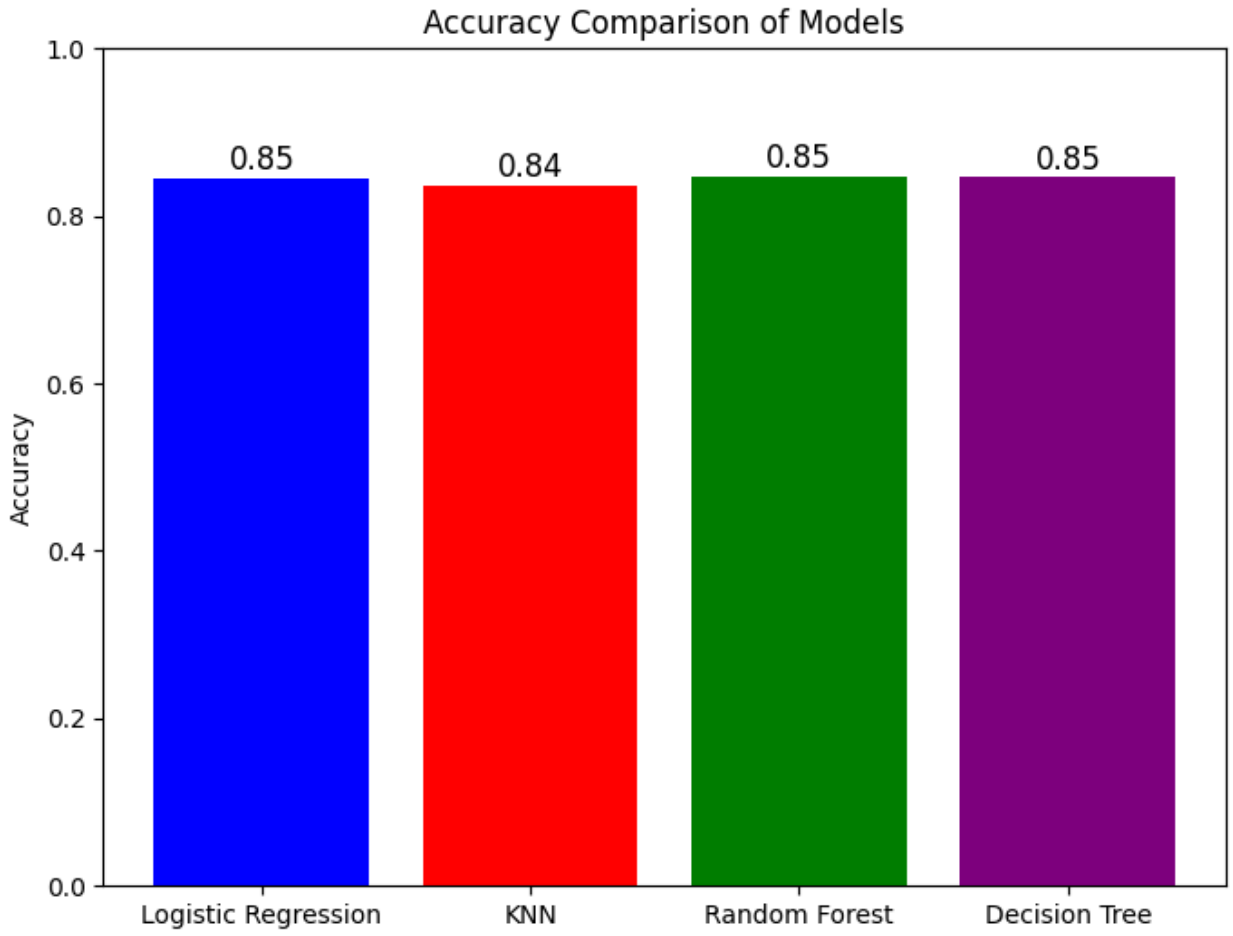
```
# KNN ROC
probs_knn = knn.predict_proba(X_test_scaled)
for i in range(3):
    fpr, tpr, _ = roc_curve(y_test_bin[:, i], probs_knn[:, i])
    plt.plot(fpr, tpr, '--.', label=f"KNN class {i}")
plt.plot([0, 1], [0, 1], 'k--')
plt.title("Multiclass ROC Curves")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```



In [26]:

```
# 11. Accuracy Comparison Bar Graph with multiple named colors
import matplotlib.pyplot as plt
import numpy as np

model_names = ["Logistic Regression", "KNN", "Random Forest", "Decision Tree"]
accuracies = [
    accuracy_score(y_test, pred_lr),
    accuracy_score(y_test, pred_knn),
    accuracy_score(y_test, pred_rf),
    accuracy_score(y_test, pred_dt)
]

# Named colors for each bar
colors = ["blue", "red", "green", "purple"]

x_pos = np.arange(len(model_names))  # positions of bars

plt.figure(figsize=(8, 6))
plt.bar(x_pos, accuracies, color=colors)
plt.xticks(x_pos, model_names)
plt.ylim(0, 1)
plt.ylabel("Accuracy")
plt.title("Accuracy Comparison of Models")

# Add accuracy values on top of bars
for i, acc in enumerate(accuracies):
```

```
    plt.text(i, acc + 0.01, f"{acc:.2f}", ha='center', fontsize=12)

plt.show()
```



In [ ]: