In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score, accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler

# Load and preprocess data
def load_and_preprocess_data(file_path):
    data = pd.read_csv(file_path)
    print("Dataset:\n")
    print(data.head(5))
    print("\nShape of the Dataset:\n")
    print(data.shape)
    data.fillna(data.mean(), inplace=True)
    data['OverallScore'] = (
        data['GPA'] * 0.4 +
        data['Hackathons'] * 2.0 +
        data['Papers'] * 1.5 +
        data['Teacher Assistance'] * 0.5 +
        data['Consistency'] * 0.2 +
        data['Extracurriculars'] * 0.3 +
        data['Internships'] * 1.5 +
        data['Leadership Roles'] * 2.0
    )
    return data

# Train and compare models, return best model
def train_and_compare_models(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

    # Keep X_test as a DataFrame to retain indices
    X_test_orig = X_test.copy()

    scaler = StandardScaler()
    X_train_scaled, X_test_scaled = scaler.fit_transform(X_train), scaler.transf

    models = {
        'RandomForest': RandomForestRegressor(n_estimators=100, random_state=42)
        'LinearRegression': LinearRegression(),
        'DecisionTree': DecisionTreeRegressor(random_state=42)
    }

    # Customize plot colors for the R^2 score comparison
    colors = ['gray']  # Customized colors (Orange, Green, Blue)
    results = {name: r2_score(y_test, model.fit(X_train_scaled, y_train).predict

    plt.bar(results.keys(), results.values(), color=colors, width=0.5)  # Adjust
    print()
    plt.title('R^2 Score Comparison with different models')
    plt.show()

    return models['RandomForest'].fit(X_train_scaled, y_train), X_test_scaled, y

# Display top features from the best model
```

```
def display_top_features(model, X):
    features = pd.DataFrame({'Feature': X.columns, 'Importance': model.feature_i

    # Customized colors for feature importance plot
    color_map = plt.cm.get_cmap('cool')  # "cool" colormap for a gradient effect
    bar_colors = color_map(np.linspace(0, 1, len(features)))

    plt.barh(features['Feature'], features['Importance'], color=bar_colors)
    plt.title('Feature Importance')
    plt.gca().invert_yaxis()
    plt.show()

# Convert scores to categories (Low, Medium, High)
def categorize_scores(y_pred):
    bins = [-np.inf, 0.4, 0.7, np.inf]  # Bins for Low, Medium, High categories
    labels = ['Low', 'Medium', 'High']
    return pd.cut(y_pred, bins=bins, labels=labels)

# Get top 3 students
def get_top_students(X_test_orig, y_pred, y_test, data):
    top_students = pd.DataFrame({'StudentID': data.loc[X_test_orig.index, 'Stude
    print()
    print(top_students)

# Main function
def main():

    data = load_and_preprocess_data('student_performance_dataset.csv')
    X, y = data.drop(columns=['StudentID', 'OverallScore']), data['OverallScore'

    model, X_test_scaled, y_test, X_test_orig = train_and_compare_models(X, y)
    print()
    print("Feature Importance Graph:")
    display_top_features(model, X)

    # Predictions
    y_pred = model.predict(X_test_scaled)

    # Show classification report and accuracy score based on categories
    y_test_cat = categorize_scores(y_test)
    y_pred_cat = categorize_scores(y_pred)
    print()
    print("Classification Report:")
    print(classification_report(y_test_cat, y_pred_cat))
    print()
    print("Accuracy Score:", accuracy_score(y_test_cat, y_pred_cat))
    print()
    print("Top 3 Best-Performing Students :")
    get_top_students(X_test_orig, y_pred, y_test, data)

if __name__ == "__main__":
    main()
```

Dataset:

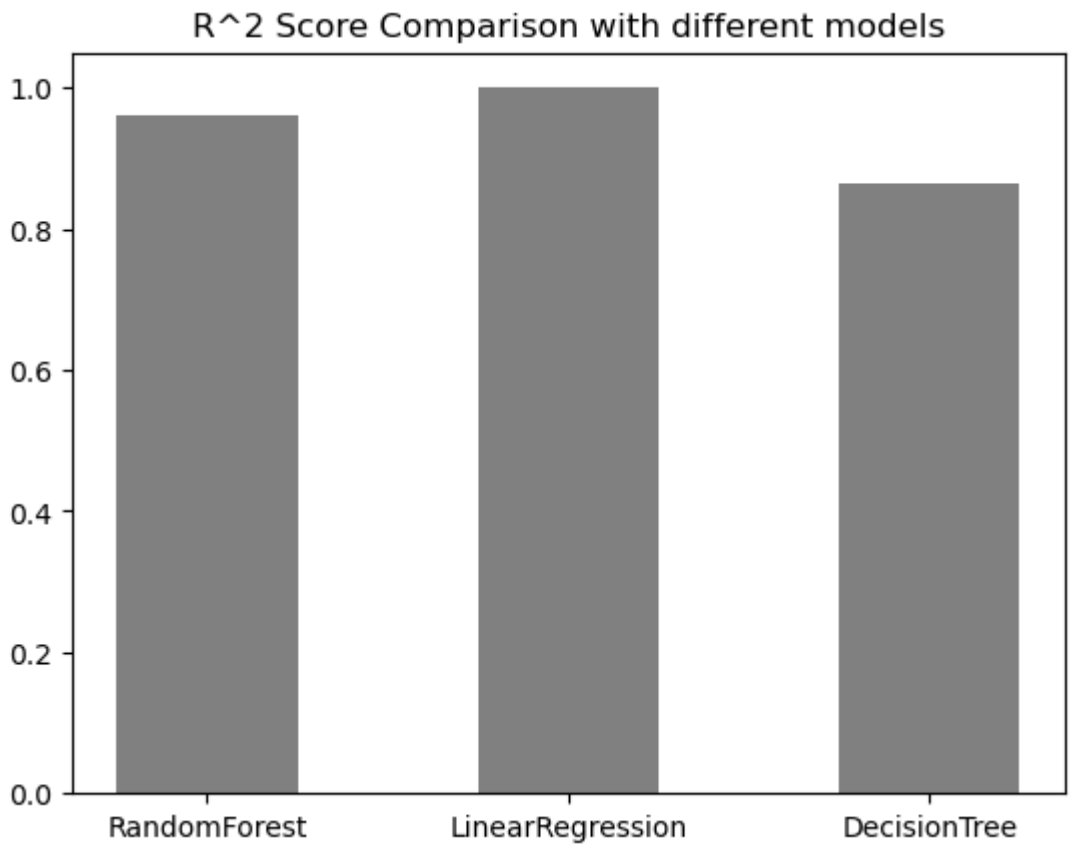
	StudentID	GPA	Hackathons	Papers	Teacher Assistance	\
0	1	6.87	2	2	0	
1	2	9.75	2	3	0	
2	3	8.66	5	0	0	
3	4	7.99	3	0	0	
4	5	5.78	2	4	1	

	Core Engineering Score	Consistency	Extracurriculars	Internships	\
0	68	74	3.57	0	
1	80	62	7.36	2	
2	87	88	6.42	3	
3	63	80	7.59	0	
4	83	61	8.20	0	

	Leadership Roles
0	0
1	1
2	1
3	1
4	1

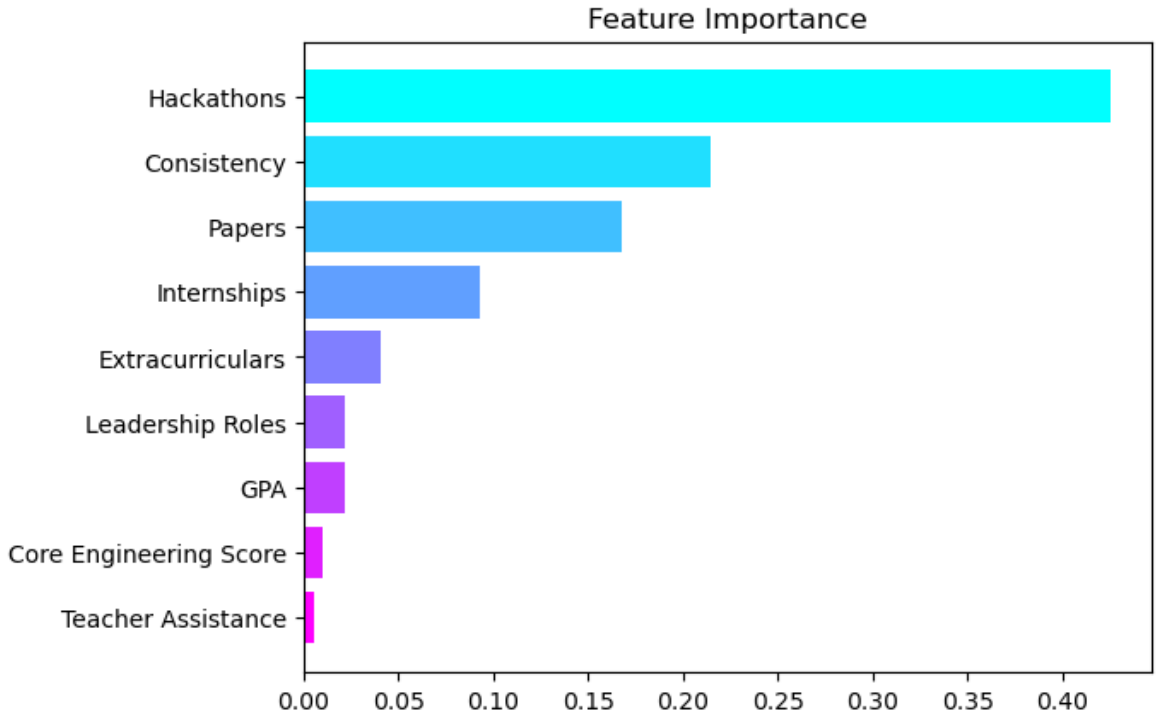
Shape of the Dataset:

(5000, 10)



Feature Importance Graph:

```
C:\Users\Sheema\AppData\Local\Temp\ipykernel_32136\1164373511.py:63: MatplotlibDe
precationWarning: The get_cmap function was deprecated in Matplotlib 3.7 and will
be removed in 3.11. Use ``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.
get_cmap()`` or ``pyplot.get_cmap()`` instead.
  color_map = plt.cm.get_cmap('cool')  # "cool" colormap for a gradient effect
```




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

        High       1.00      1.00      1.00      1000

    accuracy                           1.00      1000
   macro avg       1.00      1.00      1.00      1000
weighted avg       1.00      1.00      1.00      1000


Accuracy Score: 1.0

Top 3 Best-Performing Students :

      StudentID  PredictedScore
100         101        45.76163
3941       3942        45.15514
4686       4687        44.56010
```