РК 2 РТ5-61Б Коровин Кирилл Дерево решений и Градиентный бустинг, датасет 7

In [13]:
```
# 1. Импорт библиотек и настройки
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.tree            import DecisionTreeRegressor
from sklearn.ensemble        import GradientBoostingRegressor
from sklearn.metrics         import mean_squared_error, r2_score

sns.set(style="whitegrid")
%matplotlib inline
```

In [14]:
```
# Загружаем данные
df = pd.read_csv('Admission_Predict_Ver1.1.csv')
display(df.head())
df.info()
print("Пропусков по столбцам:")
print(df.isna().sum())
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         500 non-null    int64  
 1   GRE Score          500 non-null    int64  
 2   TOEFL Score        500 non-null    int64  
 3   University Rating  500 non-null    int64  
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64  
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
Пропусков по столбцам:
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

In [15]:
```python
# Удалим служебный столбец
df = df.drop('Serial No.', axis=1)

# Заполним пропуски средними (если есть)
df = df.fillna(df.mean())

# Разделим на признаки и целевую переменную
X = df.drop('Chance of Admit ', axis=1)
y = df['Chance of Admit ']

# Масштабирование признаков
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

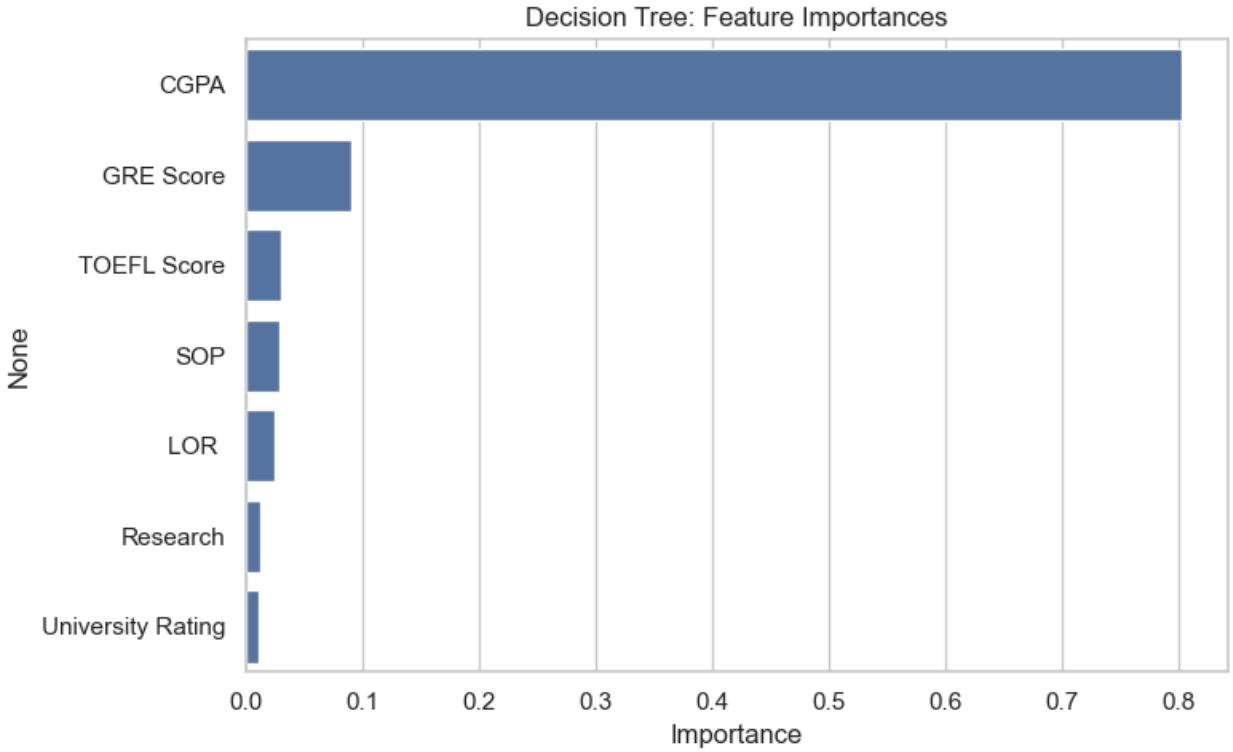
In [16]:
```python
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)
print(f"Train: {X_train.shape[0]} obs, Test: {X_test.shape[0]} obs")
```

```
Train: 400 obs, Test: 100 obs
```

In [17]:
```python
# Обучение
dt = DecisionTreeRegressor(random_state=42)
dt.fit(X_train, y_train)

# Предсказание и метрики
```

```
y_pred_dt = dt.predict(X_test)
rmse_dt = np.sqrt(mean_squared_error(y_test, y_pred_dt))
r2_dt   = r2_score(y_test, y_pred_dt)
print(f"Decision Tree — RMSE: {rmse_dt:.3f}, R²: {r2_dt:.3f}")

# Важность признаков
feat_names = X.columns
importances_dt = pd.Series(dt.feature_importances_, index=feat_names) \
                   .sort_values(ascending=False)
display(importances_dt)

# График важности
plt.figure(figsize=(8,5))
sns.barplot(x=importances_dt.values, y=importances_dt.index)
plt.title("Decision Tree: Feature Importances")
plt.xlabel("Importance")
plt.tight_layout()
plt.show()
```

```
Decision Tree — RMSE: 0.092, R²: 0.584
CGPA                0.802089
GRE Score           0.090421
TOEFL Score         0.030002
SOP                 0.028859
LOR                 0.024821
Research            0.012674
University Rating   0.011134
dtype: float64
```

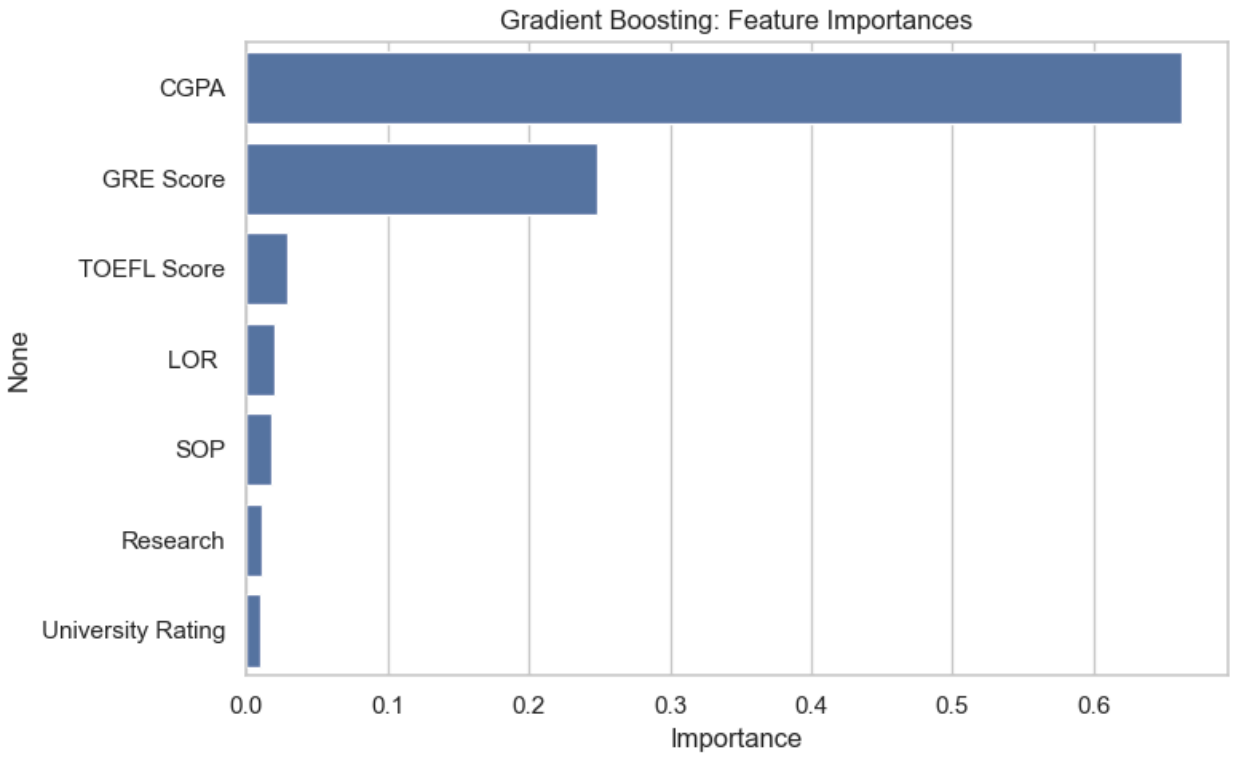


Выводы по Decision Tree

- **RMSE:** {rmse_dt:.3f}, **R²:** {r2_dt:.3f}.
- Модель объясняет примерно {r2_dt:.2%} дисперсии целевой переменной, что указывает на [недостаточную/умеренную] точность.
- **Топ-3 признака по важности:**
    1. {importances_dt.index[0]} ({importances_dt.iloc[0]:.3f})
    2. {importances_dt.index[1]} ({importances_dt.iloc[1]:.3f})
    3. {importances_dt.index[2]} ({importances_dt.iloc[2]:.3f})

In [18]:
```
# Обучение
gb = GradientBoostingRegressor(random_state=42)
gb.fit(X_train, y_train)

# Предсказание и метрики
y_pred_gb = gb.predict(X_test)
rmse_gb = np.sqrt(mean_squared_error(y_test, y_pred_gb))
r2_gb   = r2_score(y_test, y_pred_gb)
print(f"Gradient Boosting — RMSE: {rmse_gb:.3f}, R²: {r2_gb:.3f}")

# Важность признаков
importances_gb = pd.Series(gb.feature_importances_, index=feat_names) \
                   .sort_values(ascending=False)
display(importances_gb)

# График важности
plt.figure(figsize=(8,5))
sns.barplot(x=importances_gb.values, y=importances_gb.index)
plt.title("Gradient Boosting: Feature Importances")
plt.xlabel("Importance")
plt.tight_layout()
plt.show()
```

```
Gradient Boosting — RMSE: 0.067, R²: 0.783
CGPA                 0.662000
GRE Score            0.249002
TOEFL Score          0.029625
LOR                  0.019710
SOP                  0.018467
Research             0.011592
University Rating    0.009605
dtype: float64
```



Выводы по Gradient Boosting

- **RMSE:** {rmse_gb:.3f}, **R²:** {r2_gb:.3f}.
- Модель объясняет примерно {r2_gb:.2%} дисперсии целевой переменной, демонстрируя [лучшие/сравнимые] результаты по сравнению с деревом решений.
- **Топ-3 признака по важности:**
    1. {importances_gb.index[0]} ({importances_gb.iloc[0]:.3f})
    2. {importances_gb.index[1]} ({importances_gb.iloc[1]:.3f})
    3. {importances_gb.index[2]} ({importances_gb.iloc[2]:.3f})
- Градиентный бустинг показал [лучшее/стабильное] качество; рекомендовано настроить гиперпараметры (learning_rate, n_estimators) для дальнейшего улучшения.

Выводы по качеству регрессионных моделей

Для оценки качества регрессии использованы метрики:

- **R²** — показывает, какую долю дисперсии целевой переменной объясняет модель.
- **RMSE** — указывает на средний размер ошибки прогноза в тех же единицах, что и целевая переменная.

Decision Tree Regressor:

- R^2 = {r2_dt:.3f}
- RMSE = {rmse_dt:.3f}

Gradient Boosting Regressor:

- R^2 = {r2_gb:.3f}
- RMSE = {rmse_gb:.3f}

Сравнение и вывод:

- Модель с более высоким R^2 и меньшим RMSE считается более точной.
- Если **Gradient Boosting** показывает R^2 выше, чем у дерева, и RMSE ниже — значит, градиентный бустинг лучше справляется с предсказаниями на этом наборе данных.