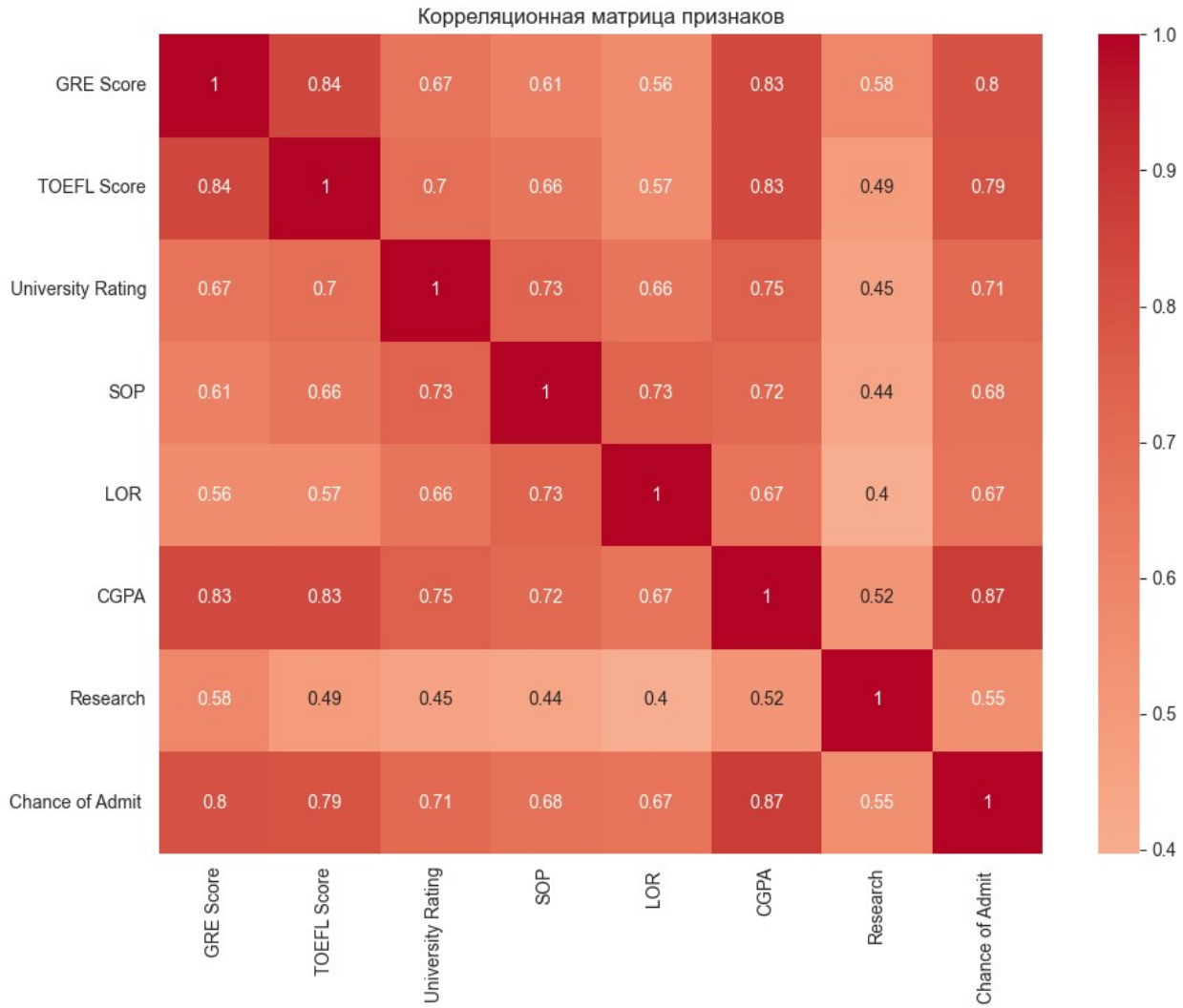
Карпенко Дмитрий Алексеевич Вариант 6 РТ5-61Б РК2 ТМО

```
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

# Настройка стиля визуализации
sns.set_style('darkgrid')
sns.set_palette('Set2')

# Загрузка данных
data = pd.read_csv('Admission_Predict.csv')

# Проверка пропусков
print("Пропуски в данных:\n", data.isnull().sum())

# Проверка имен столбцов
print("\nИмена столбцов:\n", data.columns)

# Удаление столбца 'Serial No.'
data = data.drop('Serial No.', axis=1)
```

```
Пропуски в данных:
 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

Имена столбцов:
 Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating',
'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

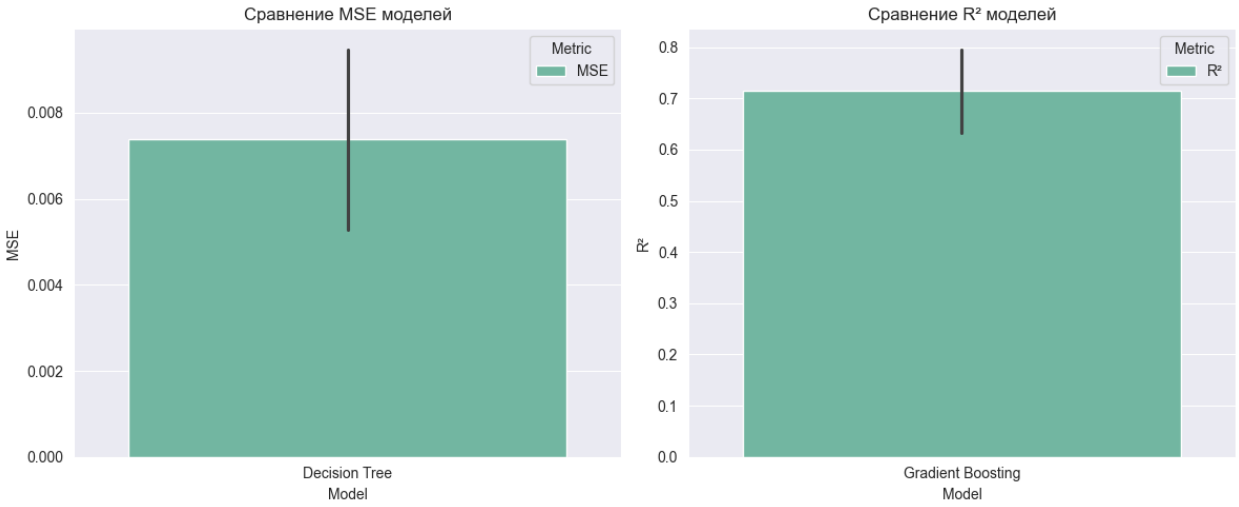
```
# Корреляционная матрица
plt.figure(figsize=(10, 8))
```

```
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', center=0)
plt.title('Корреляционная матрица признаков')
plt.tight_layout()
plt.show()

# Распределение целевой переменной (Chance of Admit)
plt.figure(figsize=(6, 4))
sns.histplot(data['Chance of Admit '], bins=20, kde=True)
plt.title('Распределение вероятности поступления')
plt.xlabel('Chance of Admit')
plt.ylabel('Частота')
plt.tight_layout()
plt.show()

# Определение признаков и целевой переменной
X = data.drop('Chance of Admit ', axis=1)
y = data['Chance of Admit ']
```




```
# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Масштабирование данных
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Инициализация моделей
models = {
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(n_estimators=100,
random_state=42)
}

# Обучение и оценка моделей
results = {}
predictions = {}
for name, model in models.items():
    try:
        # Обучение модели
        model.fit(X_train_scaled, y_train)
        # Предсказания
        y_pred = model.predict(X_test_scaled)
        # Метрики
        mse = mean_squared_error(y_test, y_pred)
```

```
        r2 = r2_score(y_test, y_pred)
        results[name] = {'MSE': mse, 'R²': r2}
        predictions[name] = y_pred
        print(f"\n{name} результаты:")
        print(f"MSE: {mse:.6f}")
        print(f"R²: {r2:.4f}")
    except Exception as e:
        print(f"Ошибка в модели {name}: {str(e)}")


Decision Tree результаты:
MSE: 0.009474
R²: 0.6331

Gradient Boosting результаты:
MSE: 0.005273
R²: 0.7958

# Сравнение метрик моделей
metrics_df = pd.DataFrame({
    'Model': [name for name in results.keys() for _ in range(2)],
    'Metric': ['MSE'] * len(results) + ['R²'] * len(results),
    'Value': [results[name]['MSE'] for name in results] +
[results[name]['R²'] for name in results]
})

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
sns.barplot(x='Model', y='Value', hue='Metric',
data=metrics_df[metrics_df['Metric'] == 'MSE'], ax=ax1)
ax1.set_title('Сравнение MSE моделей')
ax1.set_ylabel('MSE')
sns.barplot(x='Model', y='Value', hue='Metric',
data=metrics_df[metrics_df['Metric'] == 'R²'], ax=ax2)
ax2.set_title('Сравнение R² моделей')
ax2.set_ylabel('R²')
plt.tight_layout()
plt.show()
```



Какие метрики качества использованы и почему?

Для задачи регрессии я использовал две метрики качества: Mean Squared Error (MSE) и R^2 score.

Mean Squared Error (MSE):

Описание: MSE измеряет среднеквадратичную ошибку между истинными и предсказанными значениями. Она рассчитывается как среднее значение квадратов разностей между истинными и предсказанными значениями.

MSE напрямую оценивает точность предсказаний, показывая, насколько предсказанные значения отклоняются от истинных. Чем меньше MSE, тем лучше модель. Это стандартная метрика для регрессии, чувствительная к большим ошибкам (из-за квадратичной зависимости), что важно, если крупные отклонения в предсказании вероятности поступления критичны.

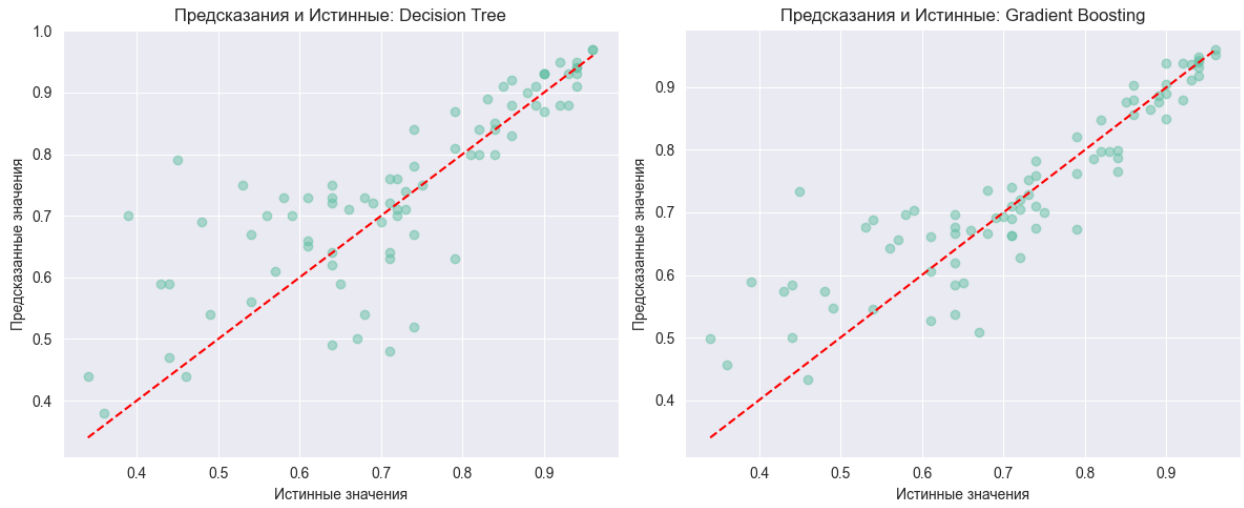
R^2 score:

Описание: R^2 (коэффициент детерминации) показывает долю дисперсии целевой переменной, объяснённую моделью. Он варьируется от 0 до 1 (или может быть отрицательным для очень плохих моделей).

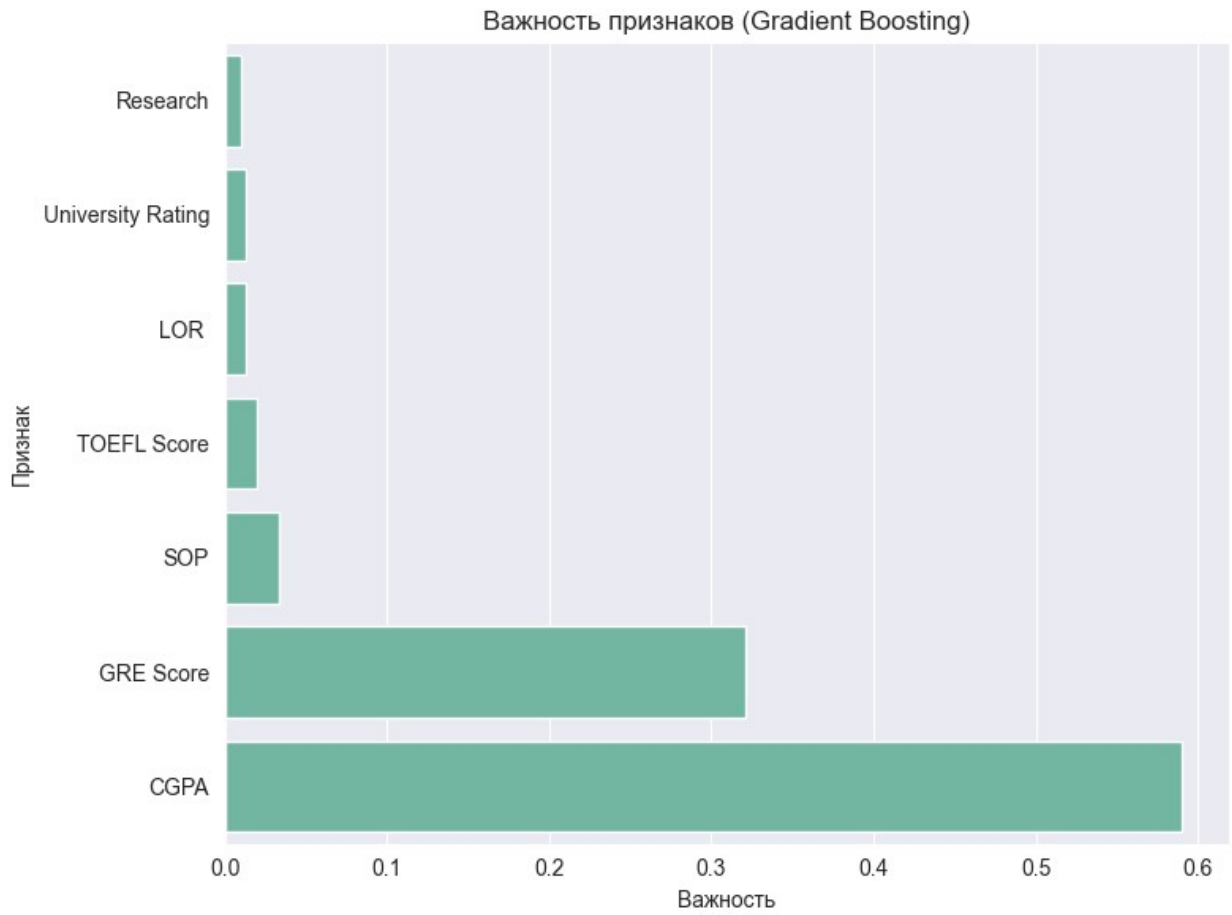
R^2 даёт представление о том, насколько модель лучше объясняет данные по сравнению с простой моделью, предсказывающей среднее значение. Значение близкое к 1 указывает на

высокую объяснительную способность. Эта метрика удобна для сравнения моделей, так как она нормализована и не зависит от масштаба данных. R^2 дополняет MSE, предоставляя относительную меру качества, тогда как MSE фокусируется на абсолютных ошибках.

```
# Предсказания и истинные значения
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
for idx, (name, y_pred) in enumerate(predictions.items()):
    axes[idx].scatter(y_test, y_pred, alpha=0.5)
    axes[idx].plot([y_test.min(), y_test.max()], [y_test.min(),
y_test.max()], 'r--')
    axes[idx].set_title(f'Предсказания и Истинные: {name}')
    axes[idx].set_xlabel('Истинные значения')
    axes[idx].set_ylabel('Предсказанные значения')
plt.tight_layout()
plt.show()
```




```
# Важность признаков для Gradient Boosting
if 'Gradient Boosting' in models:
    feature_importance = pd.DataFrame({
        'Feature': X.columns,
        'Importance': models['Gradient Boosting'].feature_importances_
    }).sort_values('Importance', ascending=True)
    plt.figure(figsize=(8, 6))
    sns.barplot(x='Importance', y='Feature', data=feature_importance)
    plt.title('Важность признаков (Gradient Boosting)')
    plt.xlabel('Важность')
    plt.ylabel('Признак')
    plt.tight_layout()
    plt.show()
```




```
# Сравнение результатов
print("\nСравнение качества моделей:")
for name, metrics in results.items():
    print(f"{name}: MSE = {metrics['MSE']:.6f}, R² =
{metrics['R²']:.4f}")
```

```
Сравнение качества моделей:
Decision Tree: MSE = 0.009474, R² = 0.6331
Gradient Boosting: MSE = 0.005273, R² = 0.7958
```