РК 2

Вариант 21
Группа РТ5-61Б
Студент: Топорин Богдан

Методы

В данной работе были использованы два метода регрессии:

- **Дерево решений** (*Decision Tree Regressor*)
- **Градиентный бустинг** (*Gradient Boosting Regressor*)

Датасет

Был использован датасет **Formula E World Championship Race Results** с платформы *Kaggle*.

In [6]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, r2_score

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

# Приводим Started и Pos к числам
data['Started'] = pd.to_numeric(data['Started'], errors='coerce')
data['Pos'] = pd.to_numeric(data['Pos'], errors='coerce')

# Чистим DriverNumber от '#'
data['DriverNumber'] = data['DriverNumber'].astype(str).str.replace('#', '', r
data['DriverNumber'] = pd.to_numeric(data['DriverNumber'], errors='coerce')

# Обработка пропусков
num_cols = ['Started', 'Pos', 'DriverNumber']
for col in num_cols:
    data[col] = data[col].fillna(data[col].median())

# Обработка категориальных признаков
# Склеиваем DriverFirstName + DriverLastName
data['DriverFullName'] = data['DriverFirstName'].astype(str) + '_' + data['Dri
```

```
categorical_cols = ['SeasonName', 'RaceName', 'DriverFullName', 'Team']
for col in categorical_cols:
    data[col] = data[col].fillna('Unknown')
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col].astype(str))

# Выбор признаков и целевой переменной
features = [
    'SeasonName', 'RaceName', 'Started', 'DriverNumber', 'DriverFullName', 'Te
]
X = data[features]
y = data['Pos']

# Деление данных
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Decision Tree
dt_params = {
    'max_depth': [5, 10, 15, 20, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

dt = DecisionTreeRegressor(random_state=42)
dt_grid = GridSearchCV(dt, dt_params, cv=5, scoring='neg_mean_absolute_error',
dt_grid.fit(X_train, y_train)

print(f"Лучшие параметры дерева: {dt_grid.best_params_}")

# Gradient Boosting
gb_params = {
    'n_estimators': [100, 200],
    'learning_rate': [0.01, 0.1],
    'max_depth': [3, 5, 7],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

gb = GradientBoostingRegressor(random_state=42)
gb_grid = GridSearchCV(gb, gb_params, cv=5, scoring='neg_mean_absolute_error',
gb_grid.fit(X_train, y_train)

print(f"Лучшие параметры градиентного бустинга: {gb_grid.best_params_}")

# Предсказания
y_pred_dt = dt_grid.predict(X_test)
y_pred_gb = gb_grid.predict(X_test)

# Оценка моделей
def evaluate_model(y_true, y_pred, model_name):
    mae = mean_absolute_error(y_true, y_pred)
```

```python
    r2 = r2_score(y_true, y_pred)
    rmse = np.sqrt(((y_true - y_pred) ** 2).mean())
    print(f"{model_name} – MAE: {mae:.3f}, RMSE: {rmse:.3f}, R2: {r2:.3f}")

evaluate_model(y_test, y_pred_dt, "Decision Tree (tuned)")
evaluate_model(y_test, y_pred_gb, "Gradient Boosting (tuned)")
```

```
Лучшие параметры дерева: {'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_s
plit': 10}
Лучшие параметры градиентного бустинга: {'learning_rate': 0.1, 'max_depth': 3,
'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100}
Decision Tree (tuned) – MAE: 4.701, RMSE: 5.792, R2: 0.033
Gradient Boosting (tuned) – MAE: 4.615, RMSE: 5.618, R2: 0.091
```