Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по рубежному контролю №2

«Методы построения моделей машинного обучения»

Вариант № 14

Выполнил:
Лупарев С.В.
группа ИУ5-63Б

Проверил:
Гапанюк Ю.Е.

Дата: 04.05.25

Подпись:

Дата:

Подпись:

Москва, 2025 г.

Задание:

Номер варианта: **14**

Номер набора данных, указанного в задаче: **14**

https://www.kaggle.com/noriuk/us-education-datasets-unification-project (файл states_all.csv)

Для заданного набора данных постройте регрессии. Для построения моделей используйте методы: Дерево решений, Случайный лес. Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Ход выполнения:

1) Загрузим данные, посмотрим количество пропусков

```
file_path = 'states_all.csv'
df = pd.read_csv(file_path)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1715 entries, 0 to 1714
Data columns (total 25 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   PRIMARY_KEY                     1715 non-null   object
 1   STATE                           1715 non-null   object
 2   YEAR                            1715 non-null   int64
 3   ENROLL                          1224 non-null   float64
 4   TOTAL_REVENUE                   1275 non-null   float64
 5   FEDERAL_REVENUE                 1275 non-null   float64
 6   STATE_REVENUE                   1275 non-null   float64
 7   LOCAL_REVENUE                   1275 non-null   float64
 8   TOTAL_EXPENDITURE               1275 non-null   float64
 9   INSTRUCTION_EXPENDITURE         1275 non-null   float64
 10  SUPPORT_SERVICES_EXPENDITURE    1275 non-null   float64
 11  OTHER_EXPENDITURE               1224 non-null   float64
 12  CAPITAL_OUTLAY_EXPENDITURE      1275 non-null   float64
 13  GRADES_PK_G                     1542 non-null   float64
 14  GRADES_KG_G                     1632 non-null   float64
 15  GRADES_4_G                      1632 non-null   float64
 16  GRADES_8_G                      1632 non-null   float64
 17  GRADES_12_G                     1632 non-null   float64
 18  GRADES_1_8_G                    1020 non-null   float64
 19  GRADES_9_12_G                   1071 non-null   float64
 20  GRADES_ALL_G                    1632 non-null   float64
 21  AVG_MATH_4_SCORE                565 non-null    float64
 22  AVG_MATH_8_SCORE                602 non-null    float64
 23  AVG_READING_4_SCORE             650 non-null    float64
 24  AVG_READING_8_SCORE             562 non-null    float64
dtypes: float64(22), int64(1), object(2)
memory usage: 335.1+ KB
```

```
print(df.isnull().sum())
print(f"\nDataset Shape: {df.shape}")
```

```
PRIMARY_KEY                       0
STATE                             0
YEAR                              0
ENROLL                          491
TOTAL_REVENUE                   440
FEDERAL_REVENUE                 440
STATE_REVENUE                   440
LOCAL_REVENUE                   440
TOTAL_EXPENDITURE               440
INSTRUCTION_EXPENDITURE         440
SUPPORT_SERVICES_EXPENDITURE    440
OTHER_EXPENDITURE               491
CAPITAL_OUTLAY_EXPENDITURE      440
GRADES_PK_G                     173
GRADES_KG_G                      83
GRADES_4_G                       83
GRADES_8_G                       83
GRADES_12_G                      83
GRADES_1_8_G                    695
GRADES_9_12_G                   644
GRADES_ALL_G                     83
AVG_MATH_4_SCORE                1150
AVG_MATH_8_SCORE                1113
AVG_READING_4_SCORE             1065
AVG_READING_8_SCORE             1153
dtype: int64

Dataset Shape: (1715, 25)
```

2) Будем предсказывать AVG_MATH_8_SCORE, удалим все столбцы, связанные с оценками, удалим все пропуски, в которых AVG_MATH_8_SCORE не определено, для остальных колонок пропуски заполним медианой для числовых и самым частым значением для категориальных. Также разделим данные на train и test.

```
# Предобработка данных
target = 'AVG_MATH_8_SCORE'
score_columns = ['AVG_MATH_4_SCORE', 'AVG_MATH_8_SCORE', 'AVG_READING_4_SCORE', 'AVG_READING_8_SCORE']
features_to_drop = ['PRIMARY_KEY', 'GRADES_1_8_G', 'GRADES_9_12_G'] + [col for col in score_columns if col != target]

df_processed = df.drop(columns=features_to_drop)
df_processed.dropna(subset=[target], inplace=True)
X = df_processed.drop(columns=[target])
y = df_processed[target]
categorical_features = ['STATE']
numerical_features = X.select_dtypes(include=np.number).columns.tolist()


numerical_pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median'))
])

categorical_pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])
# Используем ColumnTransformer из sklearn.compose
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_pipeline, numerical_features),
        ('cat', categorical_pipeline, categorical_features)
    ],
    remainder='passthrough'
)
```

```
# Разделение
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=52)
X_train_processed = preprocessor.fit_transform(X_train)
X_test_processed = preprocessor.transform(X_test)

ohe_feature_names = preprocessor.named_transformers_['cat'].get_feature_names_out(categorical_features)
all_feature_names = numerical_features + list(ohe_feature_names)
print("\nПропуски в обработанных данных для обучения:", np.isnan(X_train_processed).sum())
print("Пропуски в обработанных данных для теста:", np.isnan(X_test_processed).sum())
```

```
Пропуски в обработанных данных для обучения: 0
Пропуски в обработанных данных для теста: 0
```

3) Подберем лучшие параметры с помощью кросс-валидации для дерева и для случайного леса.

```python
# Дерево решений
dt_pipeline = Pipeline(steps=[('regressor', DecisionTreeRegressor(random_state=52))])
param_grid_dt = {
    'regressor__max_depth': [None, 10, 20, 30, 40, 50],
    'regressor__min_samples_split': [2, 5, 10, 20],
    'regressor__min_samples_leaf': [1, 3, 5, 10, 15],
    'regressor__max_features': ['sqrt', 'log2', 0.7, 1.0]
}

# Подбор параметров для дерева
grid_search_dt = GridSearchCV(dt_pipeline, param_grid_dt, cv=5,
                              scoring='neg_mean_squared_error',
                              n_jobs=-1, verbose=0)
grid_search_dt.fit(X_train_processed, y_train)

best_dt = grid_search_dt.best_estimator_
print(f"Лучшие параметры для дерева: {grid_search_dt.best_params_}")
```

```python
# Подбор параметров для дерева
grid_search_dt = GridSearchCV(dt_pipeline, param_grid_dt, cv=5,
                              scoring='neg_mean_squared_error',
                              n_jobs=-1, verbose=0)
grid_search_dt.fit(X_train_processed, y_train)

best_dt = grid_search_dt.best_estimator_
print(f"Лучшие параметры для дерева: {grid_search_dt.best_params_}")


# Случайный лес
rf_pipeline = Pipeline(steps=[('regressor', RandomForestRegressor(random_state=52, n_jobs=-1))])

param_grid_rf = {
    'regressor__n_estimators': [100, 200, 300, 400],
    'regressor__max_depth': [1, 2, 3, 5, 10],
    'regressor__min_samples_split': [2, 5, 10],
    'regressor__min_samples_leaf': [1, 3, 5],
    'regressor__max_features': ['sqrt', 'log2', 0.7, 1.0]
}


# Подбор параметров для леса
grid_search_rf = GridSearchCV(rf_pipeline, param_grid_rf, cv=5,
                              scoring='neg_mean_squared_error',
                              n_jobs=-1, verbose=0)
grid_search_rf.fit(X_train_processed, y_train)

best_rf = grid_search_rf.best_estimator_
print(f"Лучшие параметры для леса: {grid_search_rf.best_params_}")
```

Результаты: Лучшие параметры для дерева: {'regressor__max_depth': 30, 'regressor__max_features': 1.0, 'regressor__min_samples_leaf': 1, 'regressor__min_samples_split': 20}

Лучшие параметры для леса: {'regressor__max_depth': 10, 'regressor__max_features': 0.7, 'regressor__min_samples_leaf': 1, 'regressor__min_samples_split': 2, 'regressor__n_estimators': 300}

4) Запустим модели с лучшими подобранными параметрами для наших test наборов данных и посчитаем метрики.

```
# Функция оценки одной модели
def evaluate_model(name, model, X_test, y_test):
    predictions = model.predict(X_test)
    r2 = r2_score(y_test, predictions)
    mae = mean_absolute_error(y_test, predictions)
    mse = mean_squared_error(y_test, predictions)
    rmse = math.sqrt(mse)
    return {'Model': name, 'R2': r2, 'MAE': mae, 'MSE': mse, 'RMSE': rmse}

# Оценка моделей
dt_metrics = evaluate_model("Decision Tree", best_dt, X_test_processed, y_test)
rf_metrics = evaluate_model("Random Forest", best_rf, X_test_processed, y_test)

# Сводная таблица результатов
metrics_summary = pd.DataFrame([dt_metrics, rf_metrics])
metrics_summary
```

	Model	R2	MAE	MSE	RMSE
0	Decision Tree	0.686761	3.566848	25.183982	5.018364
1	Random Forest	0.697819	3.810207	24.294904	4.928986

Выводы:

Для оценки качества моделей я использовал основные 4 метрики. Результаты у моделей оказались очень близкими, значение R^2 нельзя назвать высоким, однако если мы посмотрим на MAE, то увидим, что оно немного меньше 4 для обеих моделей, при том, что значения предсказанной колонки в исходном наборе данных варьируются от 231 до 301, соответственно точность предсказаний моделей достаточно высокая.