



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»
Отчет по лабораторной работе №4
«Линейные модели, SVM и деревья решений.»

Выполнила:
студент группы ИУ5-61Б
Павловская А.А.
03.06.2021

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2021 г.

Цель лабораторной работы: изучение линейных моделей, SVM и деревьев решений

Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода `train_test_split` разделите выборку на обучающую и тестовую.
4. Обучите следующие модели:
 - одну из линейных моделей;
 - SVM;
 - дерево решений.
5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.

Набор данных: U.S. Education Datasets: Unification Project

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

Текст программы и экранные формы с примерами выполнения программы (ячейки ноутбука):

In [12]:

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_error, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.linear_model import SGDRegressor
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

Загрузка и первичный анализ данных

In [13]:

```
data = pd.read_csv('archive/states_all.csv')
```

In [14]:

```
data.head()
```

Out[14]:

	PRIMARY_KEY	STATE	YEAR	ENROLL	TOTAL_REVENUE	FEDERAL_REVENUE	STATE_REVENUE	LOCAL_REVENUE
0	1992_ALABAMA	ALABAMA	1992	NaN	2678885.0	304177.0	1659028.0	715680.0
1	1992_ALASKA	ALASKA	1992	NaN	1049591.0	106780.0	720711.0	222100.0
2	1992_ARIZONA	ARIZONA	1992	NaN	3258079.0	297888.0	1369815.0	1590376.0
3	1992_ARKANSAS	ARKANSAS	1992	NaN	1711959.0	178571.0	958785.0	574603.0
4	1992_CALIFORNIA	CALIFORNIA	1992	NaN	26260025.0	2072470.0	16546514.0	7641041.0

5 rows x 25 columns



In [15]:

```
# Список колонок с типами данных
data.dtypes
```

Out[15]:

PRIMARY_KEY	object
STATE	object
YEAR	int64
ENROLL	float64
TOTAL_REVENUE	float64
FEDERAL_REVENUE	float64
STATE_REVENUE	float64
LOCAL_REVENUE	float64

```

LOCAL_REVENUE float64
TOTAL_EXPENDITURE float64
INSTRUCTION_EXPENDITURE float64
SUPPORT_SERVICES_EXPENDITURE float64
OTHER_EXPENDITURE float64
CAPITAL_OUTLAY_EXPENDITURE float64
GRADES_PK_G float64
GRADES_KG_G float64
GRADES_4_G float64
GRADES_8_G float64
GRADES_12_G float64
GRADES_1_8_G float64
GRADES_9_12_G float64
GRADES_ALL_G float64
AVG_MATH_4_SCORE float64
AVG_MATH_8_SCORE float64
AVG_READING_4_SCORE float64
AVG_READING_8_SCORE float64
dtype: object

```

In [16]:

```

i = 0
for i in range(500,1715):
    data.drop(axis = 0,index = i, inplace = True)

```

In [17]:

```
data.shape
```

Out[17]:

```
(500, 25)
```

In [18]:

```

# Проверка наличия пустых значений
# Цикл по колонкам датасета
for col in data.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))

```

```

PRIMARY_KEY - 0
STATE - 0
YEAR - 0
ENROLL - 51
TOTAL_REVENUE - 0
FEDERAL_REVENUE - 0
STATE_REVENUE - 0
LOCAL_REVENUE - 0
TOTAL_EXPENDITURE - 0
INSTRUCTION_EXPENDITURE - 0
SUPPORT_SERVICES_EXPENDITURE - 0
OTHER_EXPENDITURE - 51
CAPITAL_OUTLAY_EXPENDITURE - 0
GRADES_PK_G - 9
GRADES_KG_G - 0
GRADES_4_G - 0
GRADES_8_G - 0
GRADES_12_G - 0
GRADES_1_8_G - 255
GRADES_9_12_G - 255
GRADES_ALL_G - 0
AVG_MATH_4_SCORE - 417
AVG_MATH_8_SCORE - 418
AVG_READING_4_SCORE - 378
AVG_READING_8_SCORE - 463

```

In [19]:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [20]:

```
# Кодирование категориального признака STATE целочисленными значениями
le = LabelEncoder()
le.fit(data.STATE)
data.STATE = le.transform(data.STATE)
```

In [21]:

```
data.drop(['YEAR', 'PRIMARY_KEY', 'AVG_MATH_4_SCORE', 'AVG_MATH_8_SCORE', 'AVG_READING_4_SCORE',
          'AVG_READING_8_SCORE', 'GRADES_1_8_G', 'GRADES_9_12_G'], axis = 1, inplace = True)
data.drop(['ENROLL', 'OTHER_EXPENDITURE'], axis = 1, inplace = True)
```

In [22]:

```
data.head()
```

Out[22]:

	STATE	TOTAL_REVENUE	FEDERAL_REVENUE	STATE_REVENUE	LOCAL_REVENUE	TOTAL_EXPENDITURE	INSTRUCTION
0	0	2678885.0	304177.0	1659028.0	715680.0	2653798.0	
1	1	1049591.0	106780.0	720711.0	222100.0	972488.0	
2	2	3258079.0	297888.0	1369815.0	1590376.0	3401580.0	
3	3	1711959.0	178571.0	958785.0	574603.0	1743022.0	
4	4	26260025.0	2072470.0	16546514.0	7641041.0	27138832.0	

In [23]:

```
data.dtypes
```

Out[23]:

```
STATE                int32
TOTAL_REVENUE        float64
FEDERAL_REVENUE      float64
STATE_REVENUE        float64
LOCAL_REVENUE        float64
TOTAL_EXPENDITURE    float64
INSTRUCTION_EXPENDITURE float64
SUPPORT_SERVICES_EXPENDITURE float64
CAPITAL_OUTLAY_EXPENDITURE float64
GRADES_PK_G          float64
GRADES_KG_G          float64
GRADES_4_G           float64
GRADES_8_G           float64
GRADES_12_G          float64
GRADES_ALL_G         float64
dtype: object
```

In [24]:

```
from sklearn.impute import SimpleImputer
```

In [25]:

```
# Обработка пропусков GRADES_PK_G
imp_num = SimpleImputer(strategy='median')
data[['GRADES_PK_G']] = imp_num.fit_transform(data[['GRADES_PK_G']])
```

In [26]:

```
# Проверка наличия пустых значений
# Цикл по колонкам датасета
```

```
for col in data.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
STATE - 0
TOTAL_REVENUE - 0
FEDERAL_REVENUE - 0
STATE_REVENUE - 0
LOCAL_REVENUE - 0
TOTAL_EXPENDITURE - 0
INSTRUCTION_EXPENDITURE - 0
SUPPORT_SERVICES_EXPENDITURE - 0
CAPITAL_OUTLAY_EXPENDITURE - 0
GRADES_PK_G - 0
GRADES_KG_G - 0
GRADES_4_G - 0
GRADES_8_G - 0
GRADES_12_G - 0
GRADES_ALL_G - 0
```

In [27]:

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

In [28]:

```
# Масштабирование данных
scl = MinMaxScaler()
data[['TOTAL_REVENUE']] = scl.fit_transform(data[['TOTAL_REVENUE']])
data[['FEDERAL_REVENUE']] = scl.fit_transform(data[['FEDERAL_REVENUE']])
data[['STATE_REVENUE']] = scl.fit_transform(data[['STATE_REVENUE']])
data[['LOCAL_REVENUE']] = scl.fit_transform(data[['LOCAL_REVENUE']])
data[['TOTAL_EXPENDITURE']] = scl.fit_transform(data[['TOTAL_EXPENDITURE']])
data[['INSTRUCTION_EXPENDITURE']] = scl.fit_transform(data[['INSTRUCTION_EXPENDITURE']])
data[['SUPPORT_SERVICES_EXPENDITURE']] = scl.fit_transform(data[['SUPPORT_SERVICES_EXPENDITURE']])
data[['GRADES_PK_G']] = scl.fit_transform(data[['GRADES_PK_G']])
data[['GRADES_KG_G']] = scl.fit_transform(data[['GRADES_KG_G']])
data[['GRADES_4_G']] = scl.fit_transform(data[['GRADES_4_G']])
data[['GRADES_8_G']] = scl.fit_transform(data[['GRADES_8_G']])
data[['GRADES_12_G']] = scl.fit_transform(data[['GRADES_12_G']])
data[['GRADES_ALL_G']] = scl.fit_transform(data[['GRADES_ALL_G']])
```

In [29]:

```
data.head()
```

Out[29]:

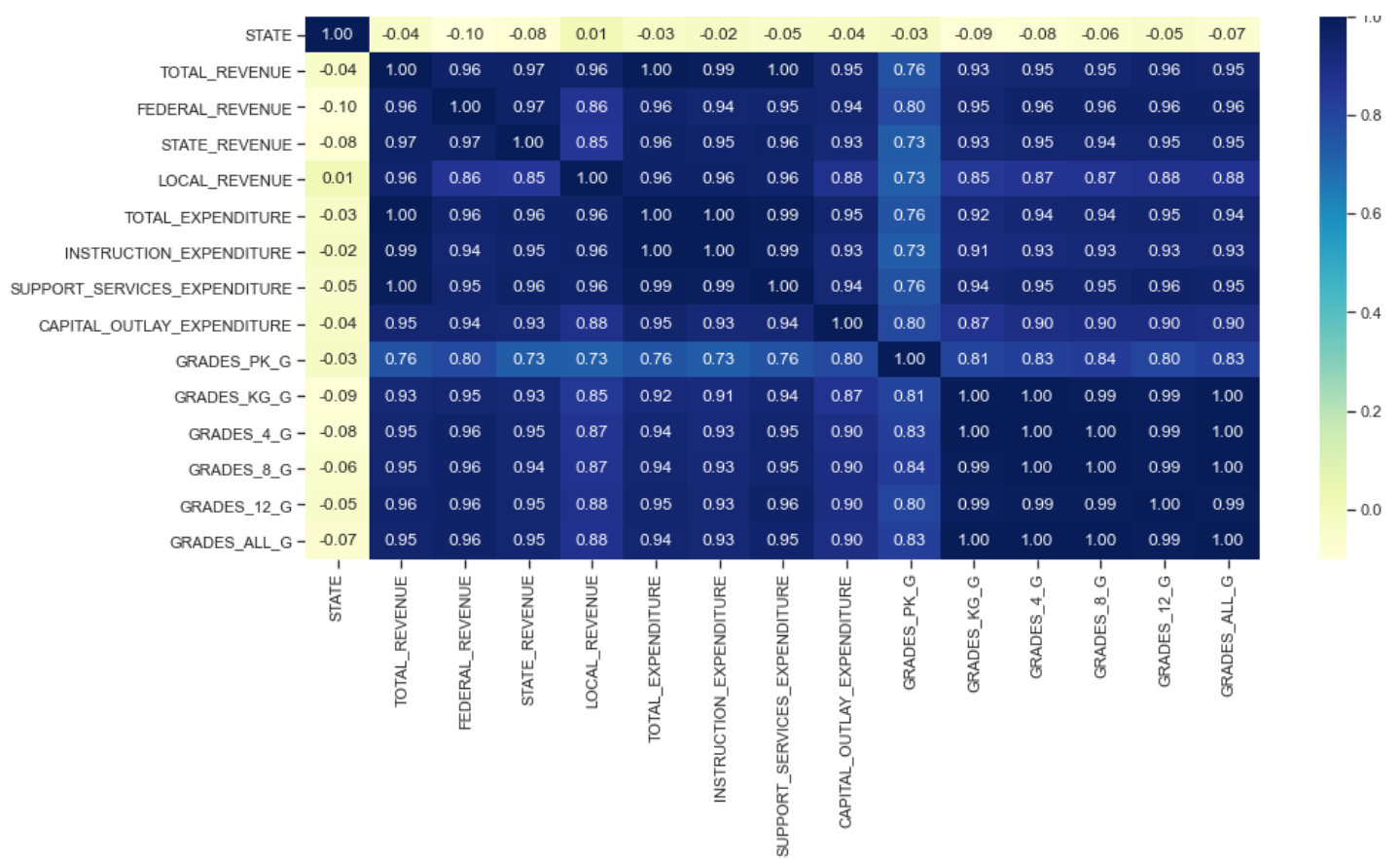
	STATE	TOTAL_REVENUE	FEDERAL_REVENUE	STATE_REVENUE	LOCAL_REVENUE	TOTAL_EXPENDITURE	INSTRUCTION
0	0	0.041899	0.064282	0.052878	0.039382	0.042095	
1	1	0.011055	0.017829	0.022971	0.011357	0.009512	
2	2	0.052864	0.062802	0.043660	0.089048	0.056587	
3	3	0.023594	0.034723	0.030559	0.031372	0.024444	
4	4	0.488314	0.480416	0.527381	0.432610	0.516603	

In [30]:

```
#Построим корреляционную матрицу
fig, ax = plt.subplots(figsize=(15,7))
sns.heatmap(data.corr(method='pearson'), cmap='YlGnBu', ax=ax, annot=True, fmt='.2f')
```

Out[30]:

<AxesSubplot:>

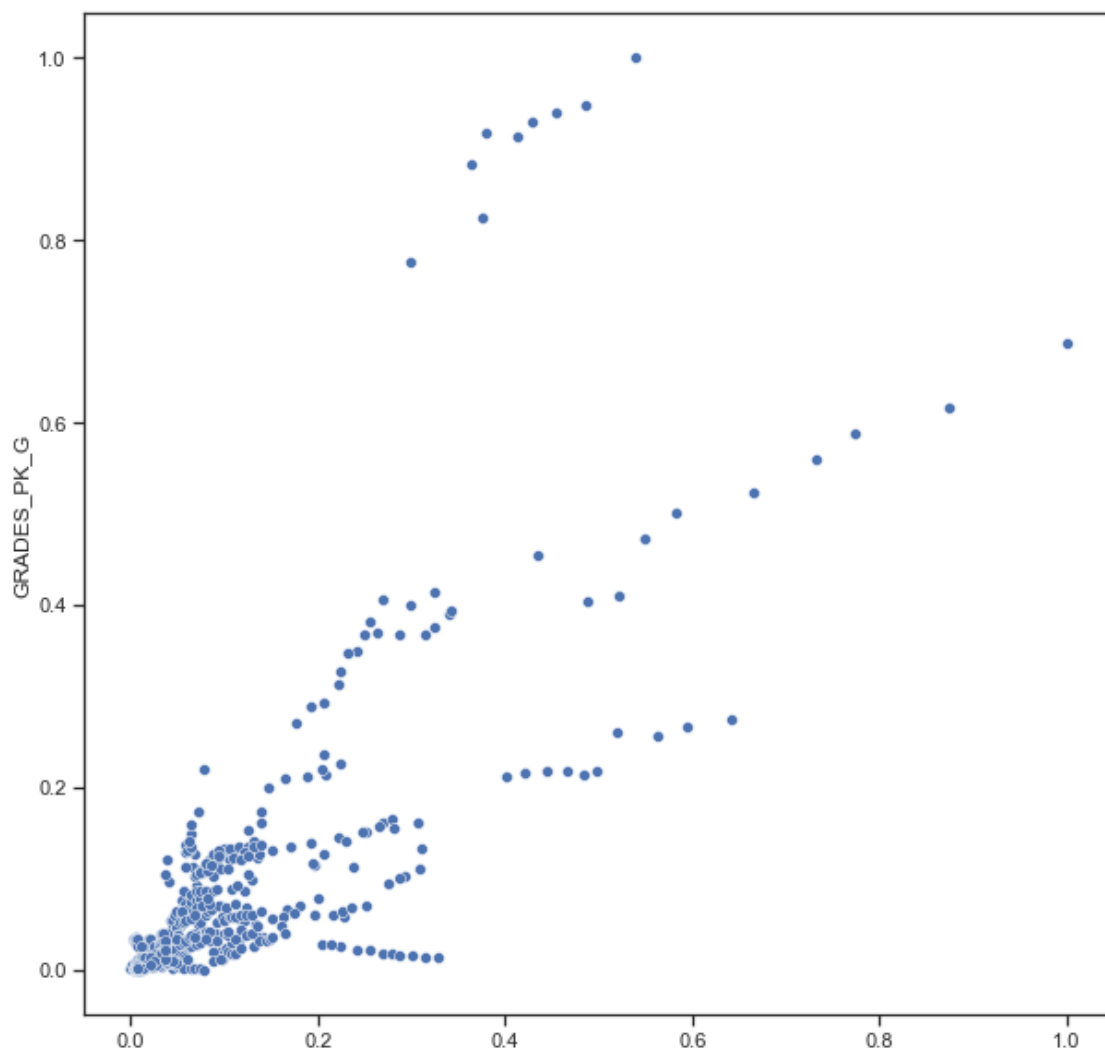


In [31]:

```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='TOTAL_REVENUE', y='GRADES_PK_G', data=data)
```

Out[31]:

<AxesSubplot:xlabel='TOTAL_REVENUE', ylabel='GRADES_PK_G'>



Между признаком **"GRADES_PK_G"** и целевым признаком **"TOTAL_REVENUE"** практически линейная зависимость, коэффициент корреляции = **0,76**

Разделение выборки на обучающую и тестовую

In [32]:

```
X = data.drop(['TOTAL_REVENUE'], axis = 1)
Y = data.TOTAL_REVENUE
print('Входные данные:\n\n', X.head(), '\n\nВыходные данные:\n\n', Y.head())
```

Входные данные:

	STATE	FEDERAL_REVENUE	STATE_REVENUE	LOCAL_REVENUE	TOTAL_EXPENDITURE \
0	0	0.064282	0.052878	0.039382	0.042095
1	1	0.017829	0.022971	0.011357	0.009512
2	2	0.062802	0.043660	0.089048	0.056587
3	3	0.034723	0.030559	0.031372	0.024444
4	4	0.480416	0.527381	0.432610	0.516603

	INSTRUCTION_EXPENDITURE	SUPPORT_SERVICES_EXPENDITURE \
0	0.046108	0.042543
1	0.008827	0.015080
2	0.044372	0.062038
3	0.026493	0.024559
4	0.534320	0.599167

	CAPITAL_OUTLAY_EXPENDITURE	GRADES_PK_G	GRADES_KG_G	GRADES_4_G \
0	174053.0	0.053417	0.107587	0.108659
1	37451.0	0.013138	0.010594	0.009072
2	609114.0	0.014328	0.103384	0.103463
3	145212.0	0.002381	0.060600	0.060486
4	2044688.0	0.403313	0.913149	0.853434

	GRADES_8_G	GRADES_12_G	GRADES_ALL_G
0	0.119394	0.106224	0.107324
1	0.011836	0.011400	0.008745
2	0.099855	0.095884	0.097912
3	0.071303	0.069024	0.060370
4	0.786271	0.737895	0.839321

Выходные данные:

```
0    0.041899
1    0.011055
2    0.052864
3    0.023594
4    0.488314
Name: TOTAL_REVENUE, dtype: float64
```

In [33]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state = 0, test_size = 0.1)
print('Входные параметры обучающей выборки:\n\n', X_train.head(), \
      '\n\nВыходные параметры тестовой выборки:\n\n', X_test.head(), \
      '\n\nВыходные параметры обучающей выборки:\n\n', Y_train.head(), \
      '\n\nВыходные параметры тестовой выборки:\n\n', Y_test.head())
```

Входные параметры обучающей выборки:

	STATE	FEDERAL_REVENUE	STATE_REVENUE	LOCAL_REVENUE	TOTAL_EXPENDITURE \
363	6	0.041782	0.066112	0.179159	0.092786
170	17	0.066298	0.070723	0.054221	0.056863
343	37	0.050512	0.070309	0.080537	0.067015
144	42	0.073162	0.054011	0.083882	0.061623
132	30	0.061741	0.098892	0.200987	0.121770

	INSTRUCTION_EXPENDITURE	SUPPORT_SERVICES_EXPENDITURE	\
363	0.104287	0.099762	
170	0.061608	0.069557	
343	0.068312	0.083281	
144	0.069587	0.061206	
132	0.123471	0.163758	

	CAPITAL_OUTLAY_EXPENDITURE	GRADES_PK_G	GRADES_KG_G	GRADES_4_G	\
363	300611.0	0.069204	0.079185	0.082541	
170	143259.0	0.136454	0.085276	0.085696	
343	319162.0	0.000585	0.069203	0.077329	
144	270608.0	0.071358	0.141738	0.128062	
132	320715.0	0.067587	0.186485	0.169457	

	GRADES_8_G	GRADES_12_G	GRADES_ALL_G
363	0.082650	0.080086	0.078576
170	0.102062	0.099701	0.095702
343	0.086189	0.089468	0.076766
144	0.135937	0.128887	0.131565
132	0.169367	0.173641	0.178946

Входные параметры тестовой выборки:

	STATE	FEDERAL_REVENUE	STATE_REVENUE	LOCAL_REVENUE	TOTAL_EXPENDITURE	\
90	39	0.003576	0.011204	0.028378	0.008137	
254	50	0.002350	0.010825	0.014838	0.003772	
283	28	0.008733	0.034713	0.029636	0.027602	
445	37	0.055902	0.078317	0.092072	0.075903	
461	2	0.129202	0.087006	0.146745	0.104198	

	INSTRUCTION_EXPENDITURE	SUPPORT_SERVICES_EXPENDITURE	\
90	0.012227	0.009161	
254	0.003585	0.004308	
283	0.022466	0.027605	
445	0.076712	0.092508	
461	0.086191	0.111668	

	CAPITAL_OUTLAY_EXPENDITURE	GRADES_PK_G	GRADES_KG_G	GRADES_4_G	\
90	12708.0	0.000021	0.013155	0.012905	
254	78444.0	0.033797	0.003380	0.003979	
283	369529.0	0.009910	0.039830	0.038386	
445	391189.0	0.001542	0.069651	0.078676	
461	1035457.0	0.042406	0.143249	0.142383	

	GRADES_8_G	GRADES_12_G	GRADES_ALL_G
90	0.015901	0.015198	0.012498
254	0.010702	0.012542	0.004953
283	0.040499	0.036358	0.036925
445	0.085182	0.094907	0.077320
461	0.144774	0.136486	0.138160

Выходные параметры обучающей выборки:

363	0.094552
170	0.057608
343	0.064866
144	0.058123
132	0.122905

Name: TOTAL_REVENUE, dtype: float64

Выходные параметры тестовой выборки:

90	0.008594
254	0.003756
283	0.023392
445	0.073902
461	0.103187

Name: TOTAL_REVENUE, dtype: float64

Линейная регрессия

In [34]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolute_error, r2_score
```

In [35]:

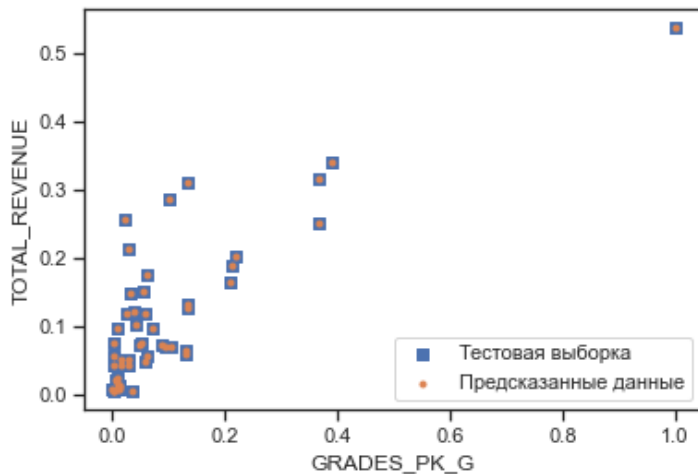
```
Lin_Reg = LinearRegression().fit(X_train, Y_train)

lr_y_pred = Lin_Reg.predict(X_test)
print('Средняя абсолютная ошибка:', mean_absolute_error(Y_test, lr_y_pred))
print('Метрика R^2 или коэффициент детерминации:', r2_score(Y_test, lr_y_pred))
```

Средняя абсолютная ошибка: 7.485023555478532e-10
Метрика R^2 или коэффициент детерминации: 0.9999999999999999

In [36]:

```
plt.scatter(X_test.GRADES_PK_G, Y_test, marker = 's', label = 'Тестовая выборка')
plt.scatter(X_test.GRADES_PK_G, lr_y_pred, marker = '.', label = 'Предсказанные данные')
plt.legend (loc = 'lower right')
plt.xlabel ('GRADES_PK_G')
plt.ylabel ('TOTAL_REVENUE')
plt.show()
```



SVM

In [37]:

```
svr_1 = SVR()
svr_1.fit(X_train, Y_train)
y_pred = svr_1.predict(X_test)
```

In [38]:

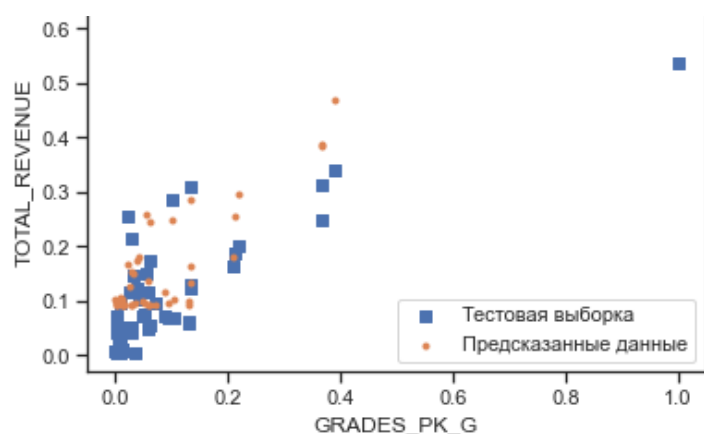
```
print('Средняя абсолютная ошибка:', mean_absolute_error(Y_test, y_pred))
print('Метрика R^2 или коэффициент детерминации:', r2_score(Y_test, y_pred))
```

Средняя абсолютная ошибка: 0.05848761044385197
Метрика R^2 или коэффициент детерминации: 0.6074477940011846

In [39]:

```
plt.scatter(X_test.GRADES_PK_G, Y_test, marker = 's', label = 'Тестовая выборка')
plt.scatter(X_test.GRADES_PK_G, y_pred, marker = '.', label = 'Предсказанные данные')
plt.legend (loc = 'lower right')
plt.xlabel ('GRADES_PK_G')
plt.ylabel ('TOTAL_REVENUE')
plt.show()
```





Дерево решений

In [40]:

```
regr_1 = DecisionTreeRegressor(max_depth=3)
regr_2 = DecisionTreeRegressor(max_depth=10)
regr_1.fit(X_train, Y_train)
regr_2.fit(X_train, Y_train)

# Предсказание
y_1 = regr_1.predict(X_test)
y_2 = regr_2.predict(X_test)
```

In [41]:

```
print('Средняя абсолютная ошибка:', mean_absolute_error(Y_test, y_1))
print('Метрика R^2 или коэффициент детерминации:', r2_score(Y_test, y_1))
print('Средняя абсолютная ошибка:', mean_absolute_error(Y_test, y_2))
print('Метрика R^2 или коэффициент детерминации:', r2_score(Y_test, y_2))
```

Средняя абсолютная ошибка: 0.015942941786388394
Метрика R^2 или коэффициент детерминации: 0.9578211419869661
Средняя абсолютная ошибка: 0.006578063970305494
Метрика R^2 или коэффициент детерминации: 0.9881525690087916

In [42]:

```
# Обучим дерево и предскажем результаты на всех признаках
tree_regr_feat_1 = DecisionTreeRegressor(random_state=1).fit(X_train, Y_train)
y_test_predict = tree_regr_feat_1.predict(X_test)
y_test_predict.shape
```

Out[42]:

(50,)

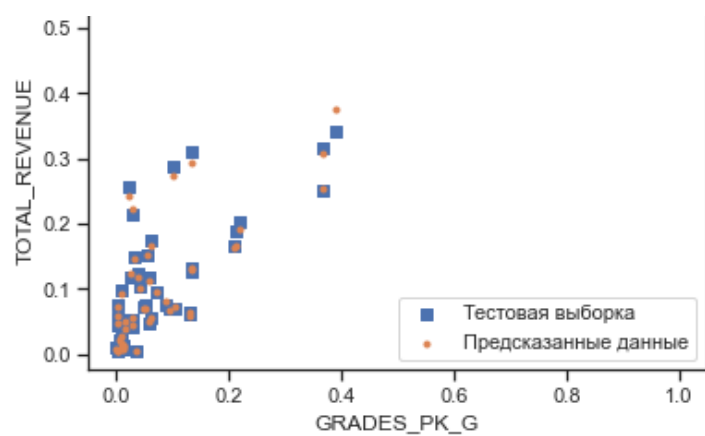
In [43]:

```
print('Средняя абсолютная ошибка:', mean_absolute_error(Y_test, y_test_predict))
print('Метрика R^2 или коэффициент детерминации:', r2_score(Y_test, y_test_predict))
```

Средняя абсолютная ошибка: 0.005265935995319338
Метрика R^2 или коэффициент детерминации: 0.9939205513271178

In [44]:

```
plt.scatter(X_test.GRADES_PK_G, Y_test, marker = 's', label = 'Тестовая выборка')
plt.scatter(X_test.GRADES_PK_G, y_test_predict, marker = '.', label = 'Предсказанные дан
ные')
plt.legend (loc = 'lower right')
plt.xlabel ('GRADES_PK_G')
plt.ylabel ('TOTAL_REVENUE')
plt.show()
```



In [11]: