Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по рубежному контролю №2

«Методы построения моделей машинного обучения.»

Вариант № 18

Выполнил:
Швецов Д.Д.
группа ИУ5-63Б

Проверил:
Гапанюк Ю.Е.

Дата: 27.05.25

Дата:

Подпись:

Подпись:

Москва, 2025 г.

Задание:

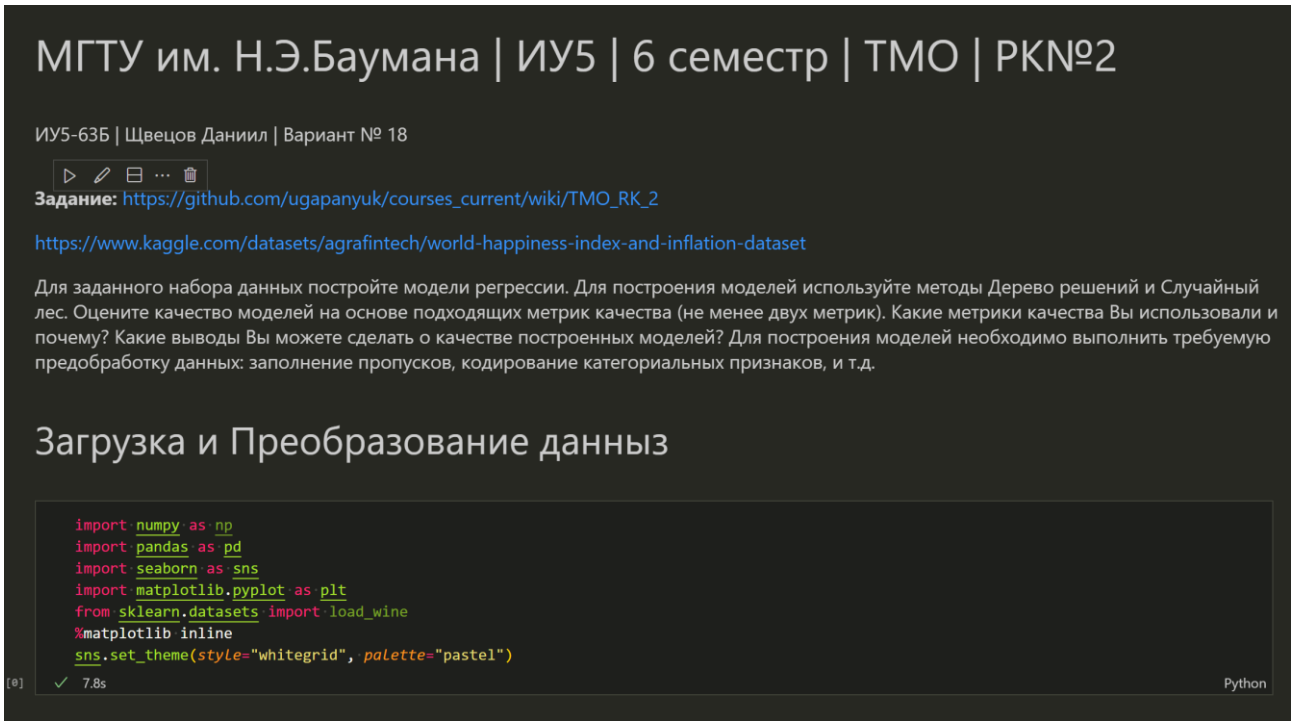
Номер варианта: **18**

Номер набора данных, указанного в задаче: **18** (https://www.kaggle.com/datasets/agrafintech/world-happiness-index-and-inflation-dataset)

Метод №1: **Дерево решений**

Метод №2: **Случайный лес**

Ход выполнения:



```
import kagglehub


path = kagglehub.dataset_download("agrafintech/world-happiness-index-and-inflation-dataset")
df0 = pd.read_csv(path+"/WHI_Inflation.csv")
df0.info()
df0.head()
```

[1] ✓ 1.0s Python

```
c:\PersonalData\university\6_sem\TMO\.conda\Lib\site-packages\tqdm\auto.py:21: TqdmWarning: IProgress not found. Please update jupyter and i
  from .autonotebook import tqdm as notebook_tqdm
Warning: Looks like you're using an outdated `kagglehub` version (installed: 0.3.11), please consider upgrading to the latest version (0.3.1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1232 entries, 0 to 1231
Data columns (total 16 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   Country                                1232 non-null   object
 1   Year                                   1232 non-null   int64
 2   Headline Consumer Price Inflation      1200 non-null   float64
 3   Energy Consumer Price Inflation        1090 non-null   float64
 4   Food Consumer Price Inflation          1130 non-null   float64
 5   Official Core Consumer Price Inflation  734 non-null   float64
 6   Producer Price Inflation               769 non-null    float64
 7   GDP deflator Index growth rate         1211 non-null   float64
 8   Continent/Region                       1232 non-null   object
 9   Score                                  1232 non-null   float64
 10  GDP per Capita                         1232 non-null   float64
 11  Social support                         1232 non-null   float64
 12  Healthy life expectancy at birth       1232 non-null   float64
 13  Freedom to make life choices           1232 non-null   float64
 14  Generosity                             1232 non-null   float64
 15  Perceptions of corruption              1231 non-null   float64
dtypes: float64(13), int64(1), object(2)
memory usage: 154.1+ KB
```

	Country	Year	Headline Consumer Price Inflation	Energy Consumer Price Inflation	Food Consumer Price Inflation	Official Core Consumer Price Inflation	Producer Price Inflation	GDP deflator Index growth rate	Continent/Region	Score	GDP per Capita	Social support	Healthy life expectancy at birth	F t
0	Afghanistan	2015	-0.660	-4.250000	-0.840000	0.219999	NaN	2.665090	South Asia	3.575	0.319820	0.302850	0.303350	
1	Afghanistan	2016	4.380	2.070000	5.670000	5.192760	NaN	-2.409509	South Asia	3.360	0.382270	0.110370	0.173440	
2	Afghanistan	2017	4.976	4.440000	6.940000	5.423228	NaN	2.404000	South Asia	3.794	0.401477	0.581543	0.180747	
3	Afghanistan	2018	0.630	1.474185	-1.045952	-0.126033	NaN	2.071208	South Asia	3.632	0.332000	0.537000	0.255000	
4	Afghanistan	2019	2.302	-2.494359	3.794770	NaN	NaN	6.520928	South Asia	3.203	0.350000	0.517000	0.361000	

```
#Заполнение пропусков
df_filled = df0.fillna(0)
```

[56] ✓ 0.0s Python

```
#Выбор стобцов
df = df_filled.drop(["Country", "Year"], axis=1)
```

[57] ✓ 0.0s Python

```
one_hot = pd.get_dummies(df['Continent/Region'], dtype=float)
df_one_hot = df.drop("Continent/Region", axis=1)
df_one_hot = df_one_hot.join(one_hot)

df_one_hot.info()
```

[9] ✓ 0.0s Python

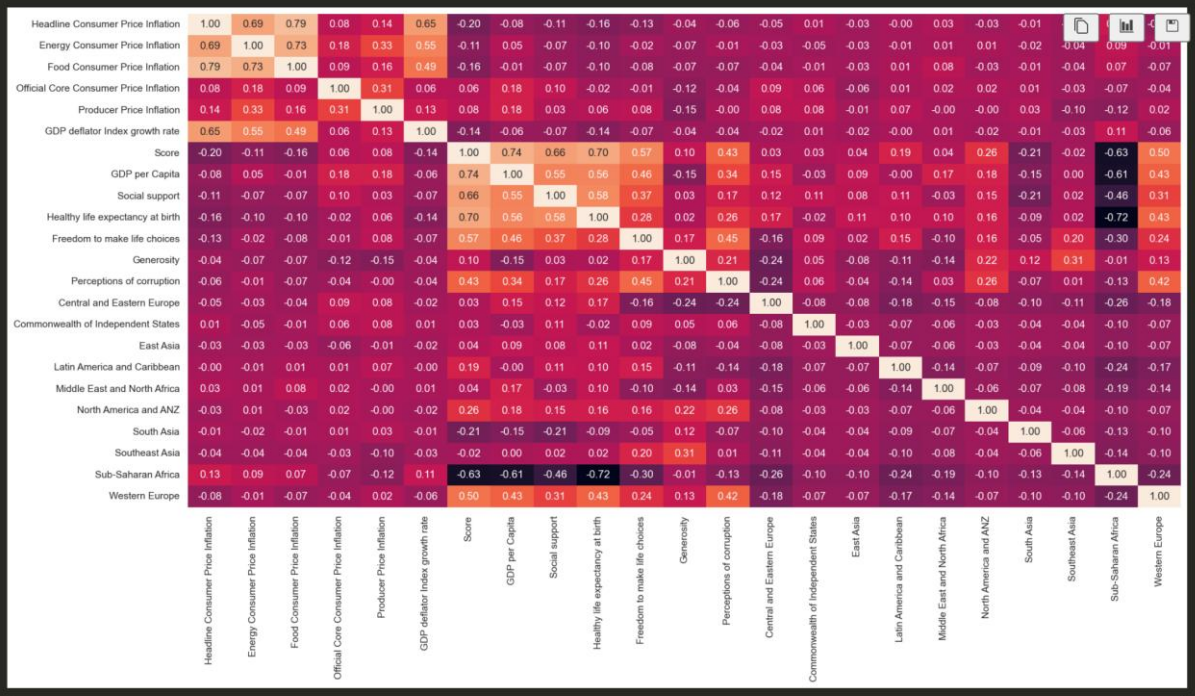
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1232 entries, 0 to 1231
Data columns (total 23 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   Headline Consumer Price Inflation      1232 non-null   float64
 1   Energy Consumer Price Inflation        1232 non-null   float64
 2   Food Consumer Price Inflation          1232 non-null   float64
 3   Official Core Consumer Price Inflation 1232 non-null   float64
 4   Producer Price Inflation               1232 non-null   float64
 5   GDP deflator Index growth rate         1232 non-null   float64
 6   Score                                  1232 non-null   float64
 7   GDP per Capita                         1232 non-null   float64
 8   Social support                         1232 non-null   float64
 9   Healthy life expectancy at birth       1232 non-null   float64
 10  Freedom to make life choices           1232 non-null   float64
 11  Generosity                             1232 non-null   float64
 12  Perceptions of corruption              1232 non-null   float64
 13  Central and Eastern Europe             1232 non-null   float64
 14  Commonwealth of Independent States     1232 non-null   float64
 15  East Asia                              1232 non-null   float64
 16  Latin America and Caribbean            1232 non-null   float64
 17  Middle East and North Africa           1232 non-null   float64
 18  North America and ANZ                  1232 non-null   float64
 19  South Asia                             1232 non-null   float64
 20  Southeast Asia                         1232 non-null   float64
 21  Sub-Saharan Africa                     1232 non-null   float64
 22  Western Europe                         1232 non-null   float64
dtypes: float64(23)
memory usage: 221.5 KB
```

Корреляционная матрица

```
fig, ax = plt.subplots(figsize=(20,10))
sns.heatmap(df_one_hot.corr(method='pearson'), ax=ax, annot=True, fmt='.2f', cbar=False)
```

[10] ✓ 0.8s Python

```
<Axes: >
```



Разбиение данных

```
from sklearn.model_selection import train_test_split


dfX = df_one_hot.drop('Score', axis=1)
dfY = df_one_hot['Score']

X_train, X_test, Y_train, Y_test = train_test_split(dfX, dfY, test_size = 0.2, random_state = 10)
```

49] ✓ 0.0s Python

Generate + Code + Markdown

```
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

50] ✓ 0.0s Python

```
(985, 22)
(247, 22)
(985,)
(247,)
```

Обучение

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor


clf = GridSearchCV(DecisionTreeRegressor(random_state=10), {'max_depth':range(3,40)})

clf.fit(X_train, Y_train)

dt_clf = clf.best_estimator_

print(clf.best_score_, clf.best_params_)
```

[51] ✓ 1.6s

```
0.7569461148115039 {'max_depth': 7}
```

```
from sklearn.ensemble import RandomForestRegressor


rf_clf = RandomForestRegressor(random_state=10, n_jobs=-1)

rf_clf.fit(X_train, Y_train)
```

[52] ✓ 0.1s

RandomForestRegressor

```
RandomForestRegressor(n_jobs=-1, random_state=10)
```

Метрики качества

средняя абсолютная ошибка

$$MAE(y, \hat{y}) = \frac{1}{N} \cdot \sum_{i=1}^{N} |y_i - \hat{y}_i|$$

где:

- y - истинное значение целевого признака
- $\hat{y}$ - предсказанное значение целевого признака
- N - размер тестовой выборки

Чем ближе значение к нулю, тем лучше качество регрессии.

коэффициент детерминации

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \overline{y_i})^2}$$

где:

- y - истинное значение целевого признака
- $\hat{y}$ - предсказанное значение целевого признака
- N - размер тестовой выборки
- $\overline{y_i} = \frac{1}{N} \cdot \sum_{i=1}^{N} y_i$

Метрики качества

средняя абсолютная ошибка

$$MAE(y, \hat{y}) = \frac{1}{N} \cdot \sum_{i=1}^{N} |y_i - \hat{y}_i|$$

где:

- y - истинное значение целевого признака
- $\hat{y}$ - предсказанное значение целевого признака
- N - размер тестовой выборки

Чем ближе значение к нулю, тем лучше качество регрессии.

коэффициент детерминации

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \overline{y_i})^2}$$

где:

- y - истинное значение целевого признака
- $\hat{y}$ - предсказанное значение целевого признака
- N - размер тестовой выборки
- $\overline{y_i} = \frac{1}{N} \cdot \sum_{i=1}^{N} y_i$

Метрика \ модель	Дерево решений	Случайный лес
MAE	0.426	0.312
R2	0.738	0.851