

**Московский государственный технический
университет им. Н. Э. Баумана**

Курс «Технологии машинного обучения»

Отчёт по рубежному контролю №2

«Технологии разведочного анализа и обработки данных.»

Вариант № 4

Выполнил:
Беспалова В.А.
группа ИУ5-63Б

Проверил:
Гапанюк Ю.Е.

Дата: 13.04.25

Дата:

Подпись:

Подпись:

Москва, 2025 г.

Полученное задание

Метод №1: Дерево решений

Метод №2: Случайный лес

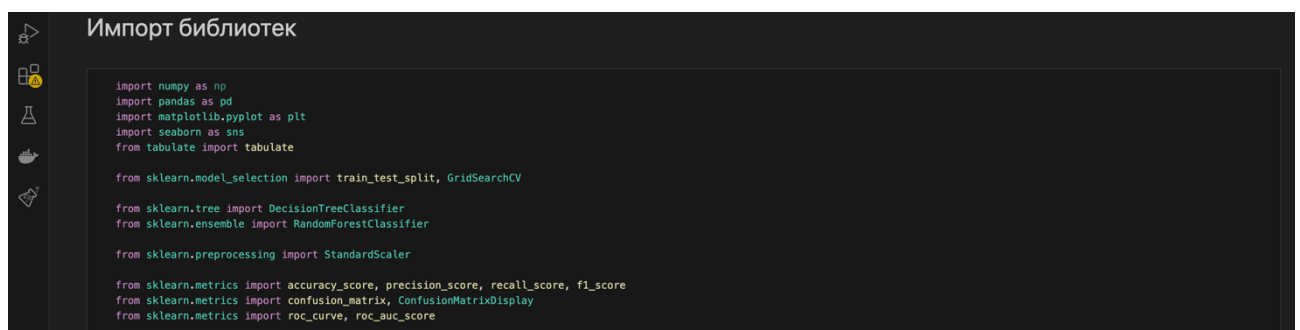
Набор данных (4): <https://www.kaggle.com/carlolepelaars/toy-dataset>

Для заданного набора данных постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы).

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



```
Импорт библиотек

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tabulate import tabulate

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import roc_curve, roc_auc_score
```

Описание полей:

Признак	Описание
Number	Индексный номер для каждой строки
City	Местоположение человека (Dallas, New York City, Los Angeles, Mountain View, Boston, Washington D.C., San Diego and Austin)
Gender	Пол человека (Male or Female)
Age	Возраст человека (в диапазоне от 25 до 65 лет).
Income	Годовой доход человека (в диапазоне от -674 до 177175).
Illness	Является ли человек больным? (Yes или No)

python

	Number	City	Gender	Age	Income	Illness
0	1	Dallas	Male	41	40367.0	No
1	2	Dallas	Male	54	45084.0	No
2	3	Dallas	Male	42	52483.0	No
3	4	Dallas	Male	40	40941.0	No
4	5	Dallas	Male	46	50289.0	No

python

(150000, 6)

python

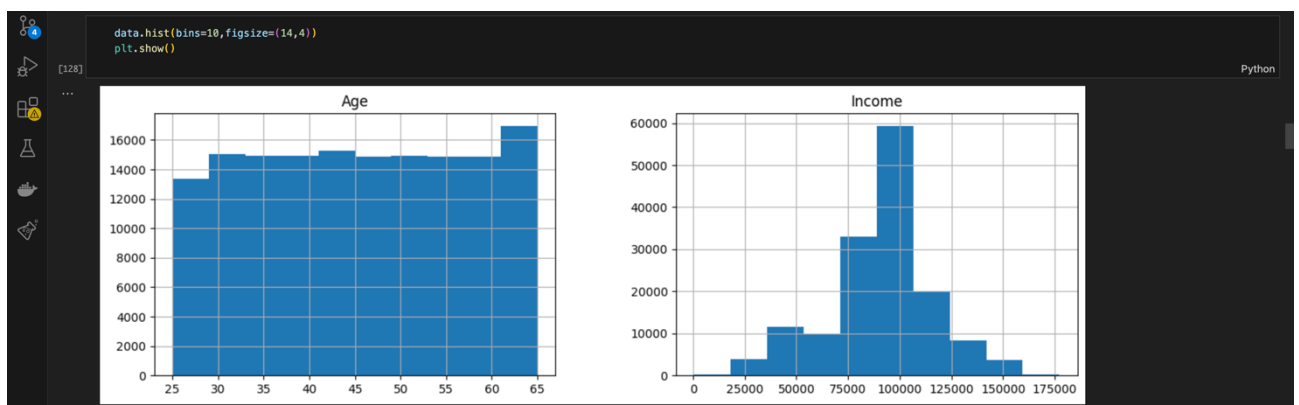
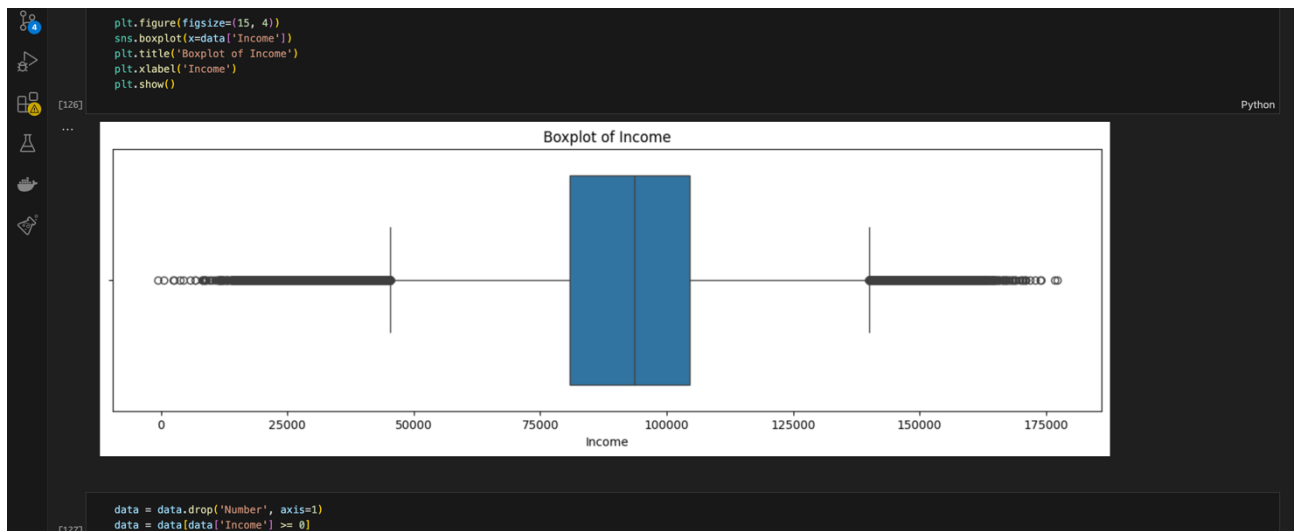
```
Number      0
City         0
Gender       0
Age          0
Income       0
Illness      0
dtype: int64
```

python

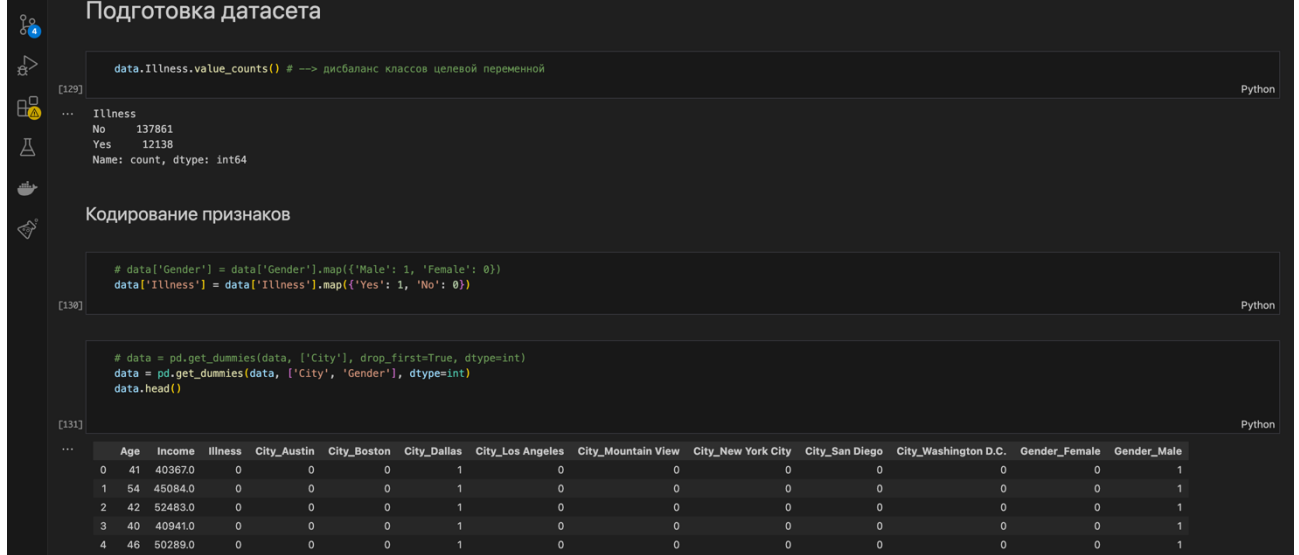
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 6 columns):
#   Column      Non-Null Count  Dtype
0   Number      150000 non-null  int64
1   City         150000 non-null  object
2   Gender       150000 non-null  object
3   Age          150000 non-null  int64
4   Income       150000 non-null  float64
5   Illness      150000 non-null  object
dtypes: float64(1), int64(2), object(3)
memory usage: 6.9+ MB
```

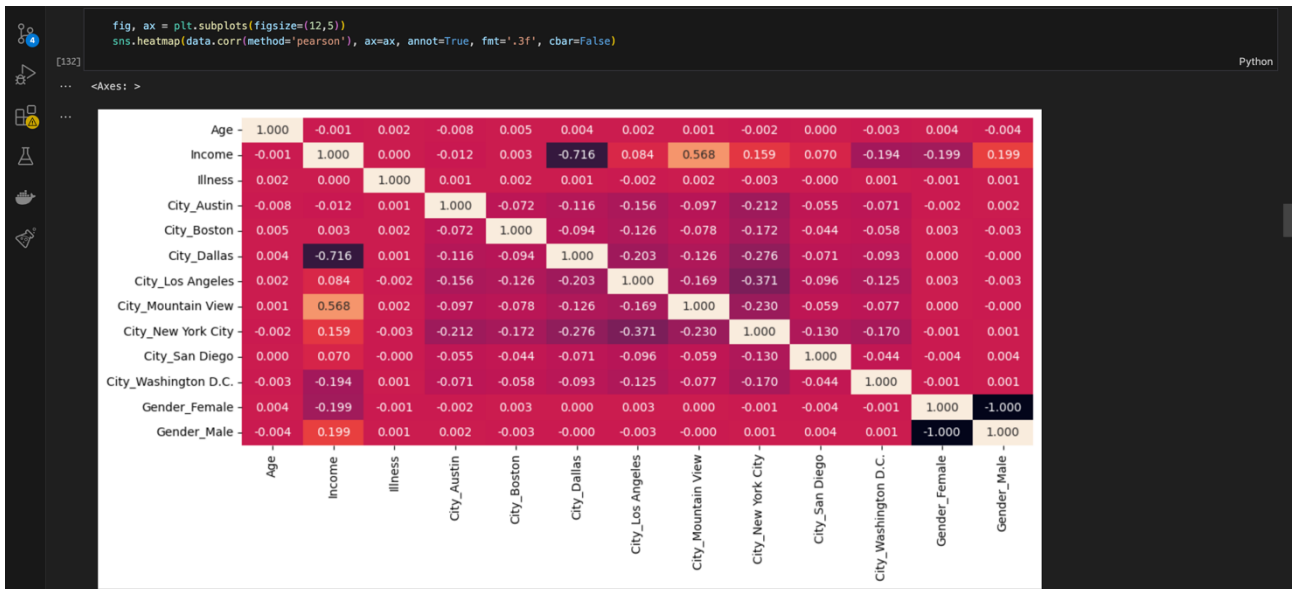
python

	Number	Age	Income
count	150000.000000	150000.000000	150000.000000
mean	75000.500000	44.950200	91252.798273
std	43301.414527	11.572486	24989.500948
min	1.000000	25.000000	-654.000000
25%	37500.750000	35.000000	80867.750000
50%	75000.500000	45.000000	93855.000000
75%	112500.250000	55.000000	104519.000000
max	150000.000000	65.000000	177157.000000



Подготовка датасета





Разделение выборки

```
X = data.drop('Illness', axis=1)
y = data['Illness']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, stratify = y, random_state = 10)
```

Python

```
# from imblearn.under_sampling import RandomUnderSampler
# undersampler = RandomUnderSampler(random_state=10)
# X_train, y_train = undersampler.fit_resample(X_train, y_train)
```

Python

Масштабирование данных

Масштабирование предполагает изменение диапазона измерения величины.

```
numerical_features = ['Age', 'Income']

scaler = StandardScaler()

X_train[numerical_features] = scaler.fit_transform(X_train[numerical_features])
X_test[numerical_features] = scaler.fit_transform(X_test[numerical_features])
```

Python

Обучение моделей

Дерево решений

```
parameters={'max_depth':range(3,30)}
clf = GridSearchCV(DecisionTreeClassifier(random_state=10, class_weight='balanced'), parameters)
clf.fit(X_train, y_train)
dt_clf = clf.best_estimator_

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

Python

0.6071224182952067 {'max_depth': 28}

Случайный лес

```
rf_clf = RandomForestClassifier(random_state=10, class_weight='balanced', n_jobs=-1)
rf_clf.fit(X_train, y_train)
```

Python

RandomForestClassifier

RandomForestClassifier(class_weight='balanced', n_jobs=-1, random_state=10)

Оценка качества моделей

Основные метрики

Accuracy
Процент (долю в диапазоне от 0 до 1) правильно определенных классов.

Precision
 $precision = \frac{TP}{TP+FP}$
Доля верно предсказанных классификатором положительных объектов, из всех объектов, которые классификатор верно или неверно определил как положительные.

Recall
 $recall = \frac{TP}{TP+FN}$
Доля верно предсказанных классификатором положительных объектов, из всех действительно положительных объектов.

F1-мера
 $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$

```

# Деревья решений
y_pred_decision_tree = dt_clf.predict(X_test)

decision_tree_accuracy = accuracy_score(y_test, y_pred_decision_tree)
decision_tree_precision = precision_score(y_test, y_pred_decision_tree, average='weighted')
decision_tree_recall = recall_score(y_test, y_pred_decision_tree, average='weighted')
decision_tree_f1 = f1_score(y_test, y_pred_decision_tree, average='weighted')

[138]
Python

# Случайный лес
y_pred_random_forest = rf_clf.predict(X_test)

random_forest_accuracy = accuracy_score(y_test, y_pred_random_forest)
random_forest_precision = precision_score(y_test, y_pred_random_forest, average='weighted')
random_forest_recall = recall_score(y_test, y_pred_random_forest, average='weighted')
random_forest_f1 = f1_score(y_test, y_pred_random_forest, average='weighted')

[139]
Python

data = [
    ['accuracy', round(decision_tree_accuracy, 3), round(random_forest_accuracy, 3)],
    ['precision', round(decision_tree_precision, 3), round(random_forest_precision, 3)],
    ['recall', round(decision_tree_recall, 3), round(random_forest_recall, 3)],
    ['f1', round(decision_tree_f1, 3), round(random_forest_f1, 3)]
]

headers = ['Метрика \ модель', 'Дерево решений', 'Случайный лес']

print(tabulate(data, headers=headers, tablefmt="grid"))

[140]
Python

...
+-----+-----+-----+
| Метрика \ модель | Деревья решений | Случайный лес |
+-----+-----+-----+
| accuracy          | 0.536           | 0.867           |
+-----+-----+-----+
| precision         | 0.852           | 0.85            |
+-----+-----+-----+
| recall            | 0.536           | 0.867           |
+-----+-----+-----+
| f1                | 0.638           | 0.858           |
+-----+-----+-----+

```



ROC-кривая и ROC AUC

Основана на вычислении следующих характеристик:

$TPR = \frac{TP}{TP+FN}$ - True Positive Rate, откладывается по оси ординат. Совпадает с recall.

$FPR = \frac{FP}{FP+TN}$ - False Positive Rate, откладывается по оси абсцисс. Показывает какую долю из объектов отрицательного класса алгоритм предсказал неверно.

В случае бинарной классификации матрица ошибок выглядит следующим образом:

Предсказанное/истинное значение	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

TPR содержит в знаменателе количество истинных 1.

FPR содержит в знаменателе количество истинных 0.

```
# Отрисовка ROC-кривой
def draw_roc_curve(y_true, y_score, pos_label, average, title):
    fpr, tpr, thresholds = roc_curve(y_true, y_score,
                                     pos_label=pos_label)
    roc_auc_value = roc_auc_score(y_true, y_score, average=average)
    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange',
             lw=lw, label='ROC curve (area = %0.2f)' % roc_auc_value)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(title)
    plt.legend(loc='lower right')
    plt.show()
```

[142]

Python

Идеальная ROC-кривая проходит через точки (0,0)-(0,1)-(1,1), то есть через верхний левый угол графика.

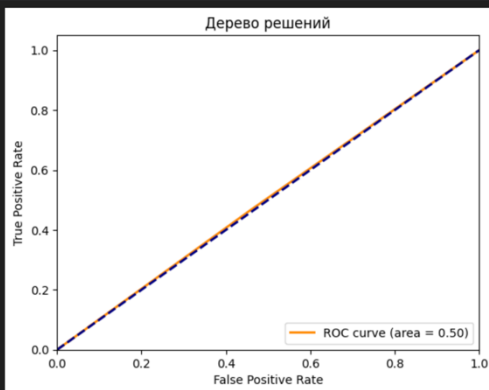
Чем сильнее отклоняется кривая от верхнего левого угла графика, тем хуже качество классификации.

В качестве количественной метрики используется площадь под кривой - ROC AUC (Area Under the Receiver Operating Characteristic Curve). Чем ниже проходит кривая тем меньше ее площадь и тем хуже качество классификатора.

```
draw_roc_curve(y_test, y_pred_decision_tree, pos_label=1, average='micro', title="Дерево решений")
```

[143]

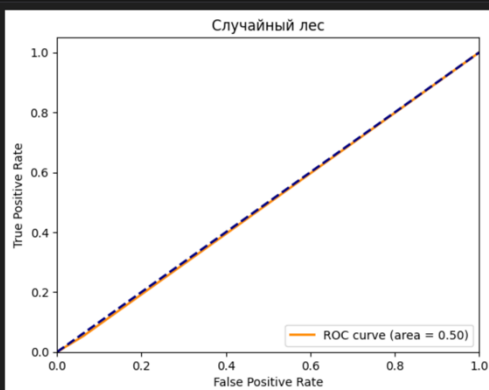
Python



```
draw_roc_curve(y_test, y_pred_random_forest, pos_label=1, average='micro', title="Случайный лес")
```

[144]

Python



+ Code

+ Markdown

Выводы

Качество моделей получилось очень низким.

В работе для оценки моделей использовались метрики для оценки качества классификации:

- accuracy, precision, recall, f1
- матрица ошибок
- ROC-кривая

Ни дерево решений, ни случайный лес не могут предсказать заболевание человека. Возможно, это связано с дисбалансом классов целевого признака.