

```
In [ ]: # han-b set kern via .ven , frmrk  
        # mglearn отстранен, тк без генер данных  
        !pip install pandas numpy matplotlib seaborn scikit-learn
```

Requirement already satisfied: pandas in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (2.3.3)

Requirement already satisfied: numpy in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (2.2.6)

Requirement already satisfied: matplotlib in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (3.10.8)

Requirement already satisfied: mglearn in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (0.2.0)

Requirement already satisfied: seaborn in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (0.13.2)

Requirement already satisfied: scikit-learn in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (1.7.2)

Requirement already satisfied: python-dateutil>=2.8.2 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from pandas) (2.9.0.post0)

Requirement already satisfied: pytz>=2020.1 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from pandas) (2025.2)

Requirement already satisfied: tzdata>=2022.7 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from pandas) (2025.3)

Requirement already satisfied: contourpy>=1.0.1 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (1.3.2)

Requirement already satisfied: cycler>=0.10 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (0.12.1)

Requirement already satisfied: fonttools>=4.22.0 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (4.61.1)

Requirement already satisfied: kiwisolver>=1.3.1 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (1.4.9)

Requirement already satisfied: packaging>=20.0 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (25.0)

Requirement already satisfied: pillow>=8 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (12.0.0)

Requirement already satisfied: pyparsing>=3 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (3.2.5)

Requirement already satisfied: imageio in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from mglearn) (2.37.2)

Requirement already satisfied: joblib in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from mglearn) (1.5.2)

Requirement already satisfied: scipy>=1.8.0 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from scikit-learn) (1.15.3)

Requirement already satisfied: threadpoolctl>=3.1.0 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from scikit-learn) (3.6.0)

Requirement already satisfied: six>=1.5 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)

1. Выбор начальных условий

а. Набор данных для задачи классификации

Реальная практическая задача:

Выявл. тенденций в мед. страховании между сборами и мед. факторами согласн kaggle-page.

- **Набор данных:**

<https://www.kaggle.com/datasets/muhammadanwaar101/healthcare-insurance-charges-dataset>

б. Набор данных для задачи регрессии

Реальная практическая задача:

Выявл. ХБП и оценки факторов риска функции поч. согласн kaggle-page.

- **Набор данных:** <https://www.kaggle.com/datasets/miadul/kidney-function-health-dataset>

с. Метрики качества и обоснование выбора

- отсылки в коммент. в коде на стр. myuller_gvido

Классификация:

- **Accuracy** - процент объектов, верно классифицированных моделью.
- **F1-score** - среднее гармоническое между точностью (precision) и полнотой (recall).

Регрессия:

- **MAE** - насколько в среднем модель ошибается в прогнозах.
- **R²** - доля дисперсии целевой переменной, которую объясняет модель. Чем ближе к 1, тем лучше.

```
In [10]: # import to get framewrk's functional.  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt
```

```
In [ ]: # 1a. upl classif datst  
# real task - factors influencing insurance charges  
clf_csv_pth = "dat_reg.csv"  
df_clf = pd.read_csv(clf_csv_pth)
```

```
In [ ]: # 1b. upl regr dat
# real task - records w biomarkers
reg_csv_pth = "dat_cl.csv"
df_reg = pd.read_csv(reg_csv_pth)
```

```
In [13]: # observ datafrs

df_clf.head()
# вывод кортеж размерн datafr
df_clf_pd = pd.DataFrame(df_clf)
display(df_clf_pd)

# # первые строк. datafr.
df_reg.head()
df_clf.shape
# вывод обобщ. данн.
df_reg.info()

# IPython.display позволяет "красиво напечатать" датафр
df_reg_pd = pd.DataFrame(df_reg)
display(df_reg_pd)

## Выбрать все строки, в которых значение столбца age больше 30
display(df_reg_pd[df_reg_pd.AGE > 20])
```

	Creatinine	BUN	GFR	Urine_Output	Diabetes	Hypertension	Age
1501	1.157595	12.883540	93.101817	1923.517271	0	0	44.0997
2586	4.795674	63.939097	32.567352	1073.853112	0	0	46.3643
2653	0.793552	18.547813	94.980260	1611.044371	0	1	28.0683
1055	0.916492	17.488186	90.292647	2146.873184	0	0	65.7616
705	1.001983	16.852746	84.319003	1757.950427	1	0	67.0666
...
1987	0.722519	7.944787	99.655555	2412.441089	0	0	44.9672
3648	0.615520	12.631735	90.221509	2218.831898	0	1	71.7788
344	0.910630	12.394649	93.024506	1526.252149	0	1	39.4233
4482	3.120306	60.703081	45.303095	1171.342076	0	1	39.7559
986	6.307465	38.855448	18.040520	971.759106	0	0	55.0813

3000 rows × 11 columns



```
<class 'pandas.core.frame.DataFrame'>
```

```
Index: 844 entries, 353 to 193
```

```
Data columns (total 7 columns):
```

#	Column	Non-Null Count	Dtype
0	AGE	844 non-null	int64
1	Gender	844 non-null	object
2	Body_Mass_Index(BMI)	844 non-null	float64
3	Number_of_Children	844 non-null	int64
4	Smoking_Status	844 non-null	object
5	Region	844 non-null	object
6	Insurance_Charges	844 non-null	float64

```
dtypes: float64(2), int64(2), object(3)
```

```
memory usage: 52.8+ KB
```

	AGE	Gender	Body_Mass_Index(BMI)	Number_of_Children	Smoking_Status	Re
353	33	male	35.245	0	no	north
864	51	male	25.400	0	no	south
1161	34	male	42.130	2	no	south
192	25	male	25.740	0	no	south
782	51	male	35.970	1	no	south
...
421	61	male	35.860	0	yes	south
164	37	male	29.640	0	no	north
28	23	male	17.385	1	no	north
516	20	male	35.310	1	no	south
193	56	female	26.600	1	no	north

844 rows × 7 columns

	AGE	Gender	Body_Mass_Index(BMI)	Number_of_Children	Smoking_Status	Re
353	33	male	35.245	0	no	north
864	51	male	25.400	0	no	south
1161	34	male	42.130	2	no	south
192	25	male	25.740	0	no	south
782	51	male	35.970	1	no	south
...
338	50	male	32.300	1	yes	north
421	61	male	35.860	0	yes	south
164	37	male	29.640	0	no	north
28	23	male	17.385	1	no	north
193	56	female	26.600	1	no	north

738 rows × 7 columns

```
In [ ]: from sklearn.model_selection import train_test_split

# для классиф. раздел. фич на катег (str_cols) и числ (num_cols).
str_cols = df_clf.select_dtypes(include=['object']).columns.tolist()

print("Cat var clf:")
print(str_cols)

str_cols_2 = df_reg.select_dtypes(include=['object']).columns.tolist()
print("Cat var reg:")
print(str_cols_2)
```

Categorical variables:

['Medication']

Categorical variables:

['Gender', 'Smoking_Status', 'Region']

2. Создание бейзлайна и оценка качества

а. Обучить модели из sklearn (для классификации и регрессии) для выбранных наборов данных

```
In [ ]: ## 2a.1 Подгот наб дан.
### 2a.1.1 раздел. датафреймовы фич на числ и катег

import matplotlib.pyplot as plt

# необх. отдел. Categorical variables: ['Medication'] и Categorical varia
# перспект.: бин.классиф. при отдел. 'Medication' и 3-ча регр.. при отдел

num_cols_clf = df_clf.drop(columns=["Medication"]).select_dtypes(include=
cat_cols_clf = df_clf.drop(columns=["Medication"]).select_dtypes(include=

num_cols_reg = df_reg.drop(columns=["Medication"]).select_dtypes(include=
cat_cols_reg = df_reg.drop(columns=["Medication"]).select_dtypes(include=
```

```
In [ ]: ### 2a.1.2 опред фич и тарг
X_clf = df_clf.drop(columns=["Medication"])
y_clf = df_clf["Medication"]

X_reg = df_reg.drop(columns=["Medication"])
y_reg = df_reg["Medication"]
```

```
In [ ]: ### 2a.1.3 разб данн согл реком книг myuller_gvido, стр. 32. rand state 0

Xclf_train, Xclf_test, yclf_train, yclf_test = train_test_split(
    X_clf, y_clf, test_size=0.25, random_state=0
)

Xreg_train, Xreg_test, yreg_train, yreg_test = train_test_split(
```

```
X_reg, y_reg, test_size=0.25, random_state=0
)
```

```
In [ ]: # learnin. basel.
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor

# clf model, k=3 from myuller_gvido, c.
knn_clf = KNeighborsClassifier(n_neighbors=3)
# Обучен. basel. модел.
knn_clf.fit(Xclf_train, yclf_train)

# clf model, also k=3
knn_reg = KNeighborsRegressor(n_neighbors=3)
knn_reg.fit(Xreg_train, yreg_train)

In [ ]: from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.neighbors import KNeighborsRegressor
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_s
from sklearn.model_selection import GridSearchCV
from collections import Counter
from sklearn.metrics import ConfusionMatrixDisplay
```

б. Оценить качество моделей (для классификации и регрессии) по выбранным метрикам на выбранных наборах данных

```
In [ ]: # 2.b. Оценить качество моделей (для классификации и регрессии) по выбран
print("Classif metr: ", acur, f_1)
print("Regr metr: ", mae, rmse, r2)

y_pred = knn.predict(X_test)
print("Прогнозы для тестового набора:\n {}".format(y_pred))

# - обосн: с 38 myulr_gvid
print("Правильность на тестовом наборе: {:.2f}".format(np.mean(y_pred ==
print("Правильность на тестовом наборе: {:.2f}".format(knn.score(X_test,
# эт. basel. оценк.
```

3. Улучшение бейзлайна

а. Сформулировать гипотезы (препроцессинг данных, визуализация данных, формирование новых признаков, подбор гиперпараметров на кросс-валидации и т.д.)

б. Проверить гипотезы

с. Сформировать улучшенный бейзлайн по результатам проверки гипотез

d. Обучить модели с улучшенным бейзлайном (для классификации и регрессии) для выбранных наборов данных

d. Обучить модели с улучшенным бейзлайном (для классификации и регрессии) для выбранных наборов данных

е. Оценить качество моделей с улучшенным бейзлайном (для классификации и регрессии) по выбранным метрикам на выбранных наборах данных

ф. Сравнить результаты моделей с улучшенным бейзлайном в сравнении с результатами из пункта 2

г. Сделать выводы

```
In [ ]: # гипот.:  
# 1. в завис. от k = 1, 3, 5 меняется точн.  
# 2. в завис. от rand state меняется кач. train_test_split  
# 3. в завис. от обработ. числ. и категор. столбц. помен. выбр. метрик.
```

```
In [ ]: # улучш бейзл  
### 2a.2 Обуч мод  
  
# для гипотезы 1:  
for k in [1, 3, 5]:  
    model = Pipeline([  
        ("prep", prep_clf),  
        ("model", KNeighborsClassifier(n_neighbors=k))  
    ])  
  
    model.fit(Xclf_train, yclf_train)  
    y_pred = model.predict(Xclf_test)  
  
    print(  
        f"k={k}, "  
        f"accuracy={accuracy_score(yclf_test, y_pred):.3f}, "  
        f"f1={f1_score(yclf_test, y_pred):.3f}"  
    )
```

```
In [ ]: # дл. гипот. 2:

for rs in [0, 21, 42]:
    X_tr, X_te, y_tr, y_te = train_test_split(
        X, y, test_size=0.2, random_state=rs
    )

    model = Pipeline([
        ("prep", prep_clf),
        ("model", KNeighborsClassifier(n_neighbors=5))
    ])

    model.fit(X_tr, y_tr)
    y_pred = model.predict(X_te)

    print(
        f"random_state={rs}, "
        f"accuracy={accuracy_score(y_te, y_pred):.3f}, "
        f"f1={f1_score(y_te, y_pred):.3f}"
    )
```

```
In [ ]: # для clf потенциально масшт. примен. прямое кодирование (при усл. дамми-
# дл. гипот. 3
prep_no_scaler = ColumnTransformer([
    ("num", "passthrough", num_cols_clf),
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols_clf)
])

prep_with_scaler = ColumnTransformer([
    ("num", StandardScaler(), num_cols_clf),
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols_clf)
])
```

```
In [ ]: # е. оценк. кач-ва модел.
mae = mean_absolute_error(yreg_test, yreg_pred)
rmse = np.sqrt(mean_squared_error(yreg_test, yreg_pred))
r2 = r2_score(yreg_test, yreg_pred)
# чтоб сдел. прогн., выз. predict объекта
yclf_pred = basel_clf.predict(Xclf_test)

# обосн выбор метр
acur = accuracy_score(yclf_test, yclf_pred)
f_1 = f1_score(yclf_test, yclf_pred)
```

г. Выводы

- согласно книге, влиян. выдвин. гипот. не крит.
- след. выдв. гипот. бол. системн.

4. Имплементация алгоритма машинного обучения

а. Самостоятельно имплементировать алгоритмы машинного обучения (для классификации и регрессии)

б. Обучить имплементированные модели (для классификации и регрессии) для выбранных наборов данных

с. Оценить качество имплементированных моделей (для классификации и регрессии) по выбранным метрикам на выбранных наборах данных

д. Сравнить результаты имплементированных моделей в сравнении с результатами из пункта 2

е. Сделать выводы

```
In [ ]: from math import sqrt
        from collections import Counter
```

```
In [ ]: # f. Добавить техники из улучшенного бейзлайна (пункт 3с)
        # внедр. в init, fit, predict
        # по книг, n_neighb = 1, допустим. и 3, и 5.

class Custom_Knn_Clf:
    def __init__(self, n_neighbors=1, metric="euclidean",
                 num_cols=None, cat_cols=None):
        self.k = n_neighbors
        self.metric = metric
        self.num_cols = num_cols
        self.cat_cols = cat_cols
        self.scaler = StandardScaler()
        self.encoder = OneHotEncoder(handle_unknown="ignore")
        self.prep_fitted = False

    # в моделях scikit-learn fit возвращает self.
    def fit(self, X, y):
        if self.num_cols or self.cat_cols:
            transformers = []
            if self.num_cols:
                transformers.append(("num", StandardScaler(), self.num_co
            if self.cat_cols:
                transformers.append(("cat", OneHotEncoder(handle_unknown=
            self.prep = ColumnTransformer(transformers)
            self.X_train = self.prep.fit_transform(X)
            self.prep_fitted = True
        else:
            self.X_train = X.values if hasattr(X, "values") else np.array
```

```

        self.y_train = np.array(y)
        return self

#  $\phi$ -ция metr, частич. замен. упрощ. sklearn

def metr(self, a, b):
    if self.metric == "euclidean":
        return math.sqrt(sum((ai - bi) ** 2 for ai, bi in zip(a, b)))
    elif self.metric == "manhattan":
        return sum(abs(ai - bi) for ai, bi in zip(a, b))
    else:
        raise ValueError("Unsupported metric")

def predict(self, X):
    if self.prep_fitted:
        X_proc = self.prep.transform(X)
    else:
        X_proc = X.values if hasattr(X, "values") else np.array(X)

    predictions = []
    for x in X_proc:
        distances = [self.metr(x, train_x) for train_x in self.X_train]
        k_idx = np.argsort(distances)[:self.k]
        k_labels = [self.y_train[i] for i in k_idx]
        most_common = Counter(k_labels).most_common(1)[0][0]
        predictions.append(most_common)
    return predictions

```

```

In [ ]: ## g. Обучить модели (для классификации и регрессии) для выбранных наборов

# улучш. clf. препроц. с k=1 и манхет. метр.
prep_clf = Pipeline([
    ("prep", prep_clf_imp),
    ("model", Custom_Knn_Clf(n_neighbors=1, metric="euclidean"))
])

prep_clf.fit(Xclf_train, yclf_train)
yclf_pred_my = prep_clf.predict(Xclf_test)

acc_my = accuracy_score(yclf_test, yclf_pred_my)
f1_my = f1_score(yclf_test, yclf_pred_my)

print(acc_my, f1_my)

# улучш. регр. препроц. с k=1 и манхет. метр.
prep_reg = Pipeline([
    ("prep", preprocessor_reg_imp),
    ("model", Custom_Knn_Reg(n_neighbors=1, metric="manhattan"))
])

# обуч. и предсказ.
prep_reg.fit(Xreg_train, yreg_train)
yreg_pred_my = prep_reg.predict(Xreg_test)

# исп. метр. класса
cust_mae = mean_absolute_error(yreg_train, yreg_pred_my)
cust_rmse = np.sqrt(mean_squared_error(yreg_train, yreg_pred_my))
cust_r2 = r2_score(yreg_train, yreg_pred_my)

```

```
print(cust_mae, cust_rmse, cust_r2)
```

```
In [ ]: # h. Оценить качество моделей (для классификации и регрессии) по выбранны
print("clf")
print("model---Accuracy---f1")
print("basel {acc:.4f}{f1:.4f}")
print("impr_metr{acc_imp:.4f}{f1_imp:.4f}")
print("knn{acc_my:.4f}{f1_my:.4f}")
```

```
In [ ]: # i. сравнить результаты моделей в сравнении с результатами из пункта 3.
print("regr")
print("mae---rmse---r²")
print("basel{mae:.2f}{rmse:.2f}{r2:.4f}")
print("impr_metr\{imp_mae:.2f}{imp_rmse:.2f}{imp_r2:.4f}")
print("custom_metr{cust_mae:.2f}{cust_rmse:.2f}{cust_r2:.4f}")
```

j. сделать выводы

- Улучш. basel. неплох.
- Предобаб. дан. знач. в работе.

```
In [ ]: # han-b set kern via .ven , frmrk  
        # mglearn отстранен, тк без генер данных  
        !pip install pandas numpy matplotlib seaborn scikit-learn
```

Requirement already satisfied: pandas in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (2.3.3)

Requirement already satisfied: numpy in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (2.2.6)

Requirement already satisfied: matplotlib in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (3.10.8)

Requirement already satisfied: mglearn in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (0.2.0)

Requirement already satisfied: seaborn in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (0.13.2)

Requirement already satisfied: scikit-learn in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (1.7.2)

Requirement already satisfied: python-dateutil>=2.8.2 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from pandas) (2.9.0.post0)

Requirement already satisfied: pytz>=2020.1 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from pandas) (2025.2)

Requirement already satisfied: tzdata>=2022.7 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from pandas) (2025.3)

Requirement already satisfied: contourpy>=1.0.1 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (1.3.2)

Requirement already satisfied: cycler>=0.10 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (0.12.1)

Requirement already satisfied: fonttools>=4.22.0 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (4.61.1)

Requirement already satisfied: kiwisolver>=1.3.1 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (1.4.9)

Requirement already satisfied: packaging>=20.0 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (25.0)

Requirement already satisfied: pillow>=8 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (12.0.0)

Requirement already satisfied: pyparsing>=3 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (3.2.5)

Requirement already satisfied: imageio in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from mglearn) (2.37.2)

Requirement already satisfied: joblib in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from mglearn) (1.5.2)

Requirement already satisfied: scipy>=1.8.0 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from scikit-learn) (1.15.3)

Requirement already satisfied: threadpoolctl>=3.1.0 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from scikit-learn) (3.6.0)

Requirement already satisfied: six>=1.5 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)

1. Выбор начальных условий

а. Набор данных для задачи классификации

Реальная практическая задача:

Выявл. тенденций в мед. страховании между сборами и мед. факторами согласн kaggle-page.

- **Набор данных:**

<https://www.kaggle.com/datasets/muhammadanwaar101/healthcare-insurance-charges-dataset>

б. Набор данных для задачи регрессии

Реальная практическая задача:

Выявл. ХБП и оценки факторов риска функции поч. согласн kaggle-page.

- **Набор данных:** <https://www.kaggle.com/datasets/miadul/kidney-function-health-dataset>

с. Метрики качества и обоснование выбора

- отсылки в коммент. в коде на стр. myuller_gvido

Классификация:

- **Accuracy** - процент объектов, верно классифицированных моделью.
- **F1-score** - среднее гармоническое между точностью (precision) и полнотой (recall).

Регрессия:

- **MAE** - насколько в среднем модель ошибается в прогнозах.
- **R²** - доля дисперсии целевой переменной, которую объясняет модель. Чем ближе к 1, тем лучше.

```
In [10]: # import to get framewrk's functional.  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt
```

```
In [ ]: # 1a. upl classif datst  
# real task - factors influencing insurance charges  
clf_csv_pth = "dat_reg.csv"  
df_clf = pd.read_csv(clf_csv_pth)
```

```
In [ ]: # 1b. upl regr dat
# real task - records w biomarkers
reg_csv_pth = "dat_cl.csv"
df_reg = pd.read_csv(reg_csv_pth)
```

```
In [13]: # observ datafrs

df_clf.head()
# вывод кортеж размерн datafr
df_clf_pd = pd.DataFrame(df_clf)
display(df_clf_pd)

# # первые строк. datafr.
df_reg.head()
df_clf.shape
# вывод обобщ. данн.
df_reg.info()

# IPython.display позволяет "красиво напечатать" датафр
df_reg_pd = pd.DataFrame(df_reg)
display(df_reg_pd)

## Выбрать все строки, в которых значение столбца age больше 30
display(df_reg_pd[df_reg_pd.AGE > 20])
```

	Creatinine	BUN	GFR	Urine_Output	Diabetes	Hypertension	Age
1501	1.157595	12.883540	93.101817	1923.517271	0	0	44.0997
2586	4.795674	63.939097	32.567352	1073.853112	0	0	46.3643
2653	0.793552	18.547813	94.980260	1611.044371	0	1	28.0683
1055	0.916492	17.488186	90.292647	2146.873184	0	0	65.7616
705	1.001983	16.852746	84.319003	1757.950427	1	0	67.0666
...
1987	0.722519	7.944787	99.655555	2412.441089	0	0	44.9672
3648	0.615520	12.631735	90.221509	2218.831898	0	1	71.7788
344	0.910630	12.394649	93.024506	1526.252149	0	1	39.4233
4482	3.120306	60.703081	45.303095	1171.342076	0	1	39.7559
986	6.307465	38.855448	18.040520	971.759106	0	0	55.0813

3000 rows × 11 columns



```
<class 'pandas.core.frame.DataFrame'>
```

```
Index: 844 entries, 353 to 193
```

```
Data columns (total 7 columns):
```

#	Column	Non-Null Count	Dtype
0	AGE	844 non-null	int64
1	Gender	844 non-null	object
2	Body_Mass_Index(BMI)	844 non-null	float64
3	Number_of_Children	844 non-null	int64
4	Smoking_Status	844 non-null	object
5	Region	844 non-null	object
6	Insurance_Charges	844 non-null	float64

```
dtypes: float64(2), int64(2), object(3)
```

```
memory usage: 52.8+ KB
```

	AGE	Gender	Body_Mass_Index(BMI)	Number_of_Children	Smoking_Status	Re
353	33	male	35.245	0	no	north
864	51	male	25.400	0	no	south
1161	34	male	42.130	2	no	south
192	25	male	25.740	0	no	south
782	51	male	35.970	1	no	south
...
421	61	male	35.860	0	yes	south
164	37	male	29.640	0	no	north
28	23	male	17.385	1	no	north
516	20	male	35.310	1	no	south
193	56	female	26.600	1	no	north

844 rows × 7 columns

	AGE	Gender	Body_Mass_Index(BMI)	Number_of_Children	Smoking_Status	Re
353	33	male	35.245	0	no	north
864	51	male	25.400	0	no	south
1161	34	male	42.130	2	no	south
192	25	male	25.740	0	no	south
782	51	male	35.970	1	no	south
...
338	50	male	32.300	1	yes	north
421	61	male	35.860	0	yes	south
164	37	male	29.640	0	no	north
28	23	male	17.385	1	no	north
193	56	female	26.600	1	no	north

738 rows × 7 columns

```
In [ ]: from sklearn.model_selection import train_test_split

# для классиф. раздел. фич на катег (str_cols) и числ (num_cols).
str_cols = df_clf.select_dtypes(include=['object']).columns.tolist()

print("Cat var clf:")
print(str_cols)

str_cols_2 = df_reg.select_dtypes(include=['object']).columns.tolist()
print("Cat var reg:")
print(str_cols_2)
```

Categorical variables:

['Medication']

Categorical variables:

['Gender', 'Smoking_Status', 'Region']

2. Создание бейзлайна и оценка качества

а. Обучить модели из sklearn (для классификации и регрессии) для выбранных наборов данных

```
In [ ]: ## 2a.1 Подгот наб дан.
### 2a.1.1 раздел. датафреймовы фич на числ и катег. повт. из лр1

import matplotlib.pyplot as plt

# необх. отдел. Categorical variables: ['Medication'] и Categorical varia
# перспект.: бин.классиф. при отдел. 'Medication' и 3-ча регр.. при отдел

num_cols_clf = df_clf.drop(columns=["Medication"]).select_dtypes(include=
cat_cols_clf = df_clf.drop(columns=["Medication"]).select_dtypes(include=

num_cols_reg = df_reg.drop(columns=["Medication"]).select_dtypes(include=
cat_cols_reg = df_reg.drop(columns=["Medication"]).select_dtypes(include=
```

```
In [ ]: ### 2a.1.2 опред фич и тарг, повт. лр.1
X_clf = df_clf.drop(columns=["Medication"])
y_clf = df_clf["Medication"]

X_reg = df_reg.drop(columns=["Medication"])
y_reg = df_reg["Medication"]
```

```
In [ ]: ### 2a.1.3 разб данн согл реком книг myuller_gvido, стр. 32. rand state 0

Xclf_train, Xclf_test, yclf_train, yclf_test = train_test_split(
    X_clf, y_clf, test_size=0.25, random_state=0
)

Xreg_train, Xreg_test, yreg_train, yreg_test = train_test_split(
    X_reg, y_reg, test_size=0.25, random_state=0
)
```

```
In [ ]: # learnin. basel. лог-рег. и лин-рег.
from sklearn.linear_model import LogisticRegression, LinearRegression

# log-reg.
log_reg = LogisticRegression(random_state=0, max_iter=500)
log_reg.fit(Xclf_train, yclf_train)

# lin-reg-task.
lin_reg = LinearRegression()
lin_reg.fit(Xreg_train, yreg_train)
```

```
In [ ]: from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.neighbors import KNeighborsRegressor
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
from collections import Counter
from sklearn.metrics import ConfusionMatrixDisplay
```

б. Оценить качество моделей (для классификации и регрессии) по выбранным метрикам на выбранных наборах данных

```
In [ ]: # 2.b. Оценить качество моделей (для классификации и регрессии) по выбранным метрикам.

# logr. basel. оценк. посл. predict и обобщ. метрик.
y_clf_pred_lr = log_reg.predict(Xclf_test)

print("logreg")

# linr-basel. оценк. посл. predict и обобщ. метрик.
y_reg_pred_lr = lin_reg.predict(Xreg_test)

print("linreg.")
```

3. Улучшение бейзлайна

а. Сформулировать гипотезы (препроцессинг данных, визуализация данных, формирование новых признаков, подбор гиперпараметров на кросс-валидации и т.д.)

б. Проверить гипотезы

с. Сформировать улучшенный бейзлайн по результатам проверки гипотез

d. Обучить модели с улучшенным бейзлайном (для классификации и регрессии) для выбранных наборов данных

d. Обучить модели с улучшенным бейзлайном (для классификации и регрессии) для выбранных наборов данных

е. Оценить качество моделей с улучшенным бейзлайном (для классификации и регрессии) по выбранным метрикам на выбранных наборах данных

ф. Сравнить результаты моделей с улучшенным бейзлайном в сравнении с результатами из пункта 2

г. Сделать выводы

```
In [ ]: # гипот.:  
# 1. в завис. от k = 1, 3, 5 меняется точн.  
# 2. в завис. от rand state меняется кач. train_test_split  
# 3. в завис. от обработ. числ. и категор. столбц. помен. выбр. метрик.
```

```
In [ ]: # улучш бейзл  
### 2a.2 Обуч мод  
  
# для гипотезы 1:  
for k in [1, 3, 5]:  
    model = Pipeline([  
        ("prep", prep_clf),  
        ("model", KNeighborsClassifier(n_neighbors=k))  
    ])  
  
    model.fit(Xclf_train, yclf_train)  
    y_pred = model.predict(Xclf_test)  
  
    print(  
        f"k={k}, "  
        f"accuracy={accuracy_score(yclf_test, y_pred):.3f}, "  
        f"f1={f1_score(yclf_test, y_pred):.3f}"  
    )
```

```
In [ ]: # дл. гипот. 2:  
  
for rs in [0, 21, 42]:  
    X_tr, X_te, y_tr, y_te = train_test_split(  
        X, y, test_size=0.2, random_state=rs  
    )
```

```

model = Pipeline([
    ("prep", prep_clf),
    ("model", KNeighborsClassifier(n_neighbors=5))
])

model.fit(X_tr, y_tr)
y_pred = model.predict(X_te)

print(
    f"random_state={rs}, "
    f"accuracy={accuracy_score(y_te, y_pred):.3f}, "
    f"f1={f1_score(y_te, y_pred):.3f}"
)

```

```

In [ ]: # для clf потенциально масшт. примен. прямое кодирование (при усл. дамми-
# дл. гипот. 3
prep_no_scaler = ColumnTransformer([
    ("num", "passthrough", num_cols_clf),
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols_clf)
])

prep_with_scaler = ColumnTransformer([
    ("num", StandardScaler(), num_cols_clf),
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols_clf)
])

```

```

In [ ]: # е. оценк. кач-ва модел.
mae = mean_absolute_error(yreg_test, yreg_pred)
rmse = np.sqrt(mean_squared_error(yreg_test, yreg_pred))
r2 = r2_score(yreg_test, yreg_pred)
# чтоб сдел. прогн., выз. predict объекта
yclf_pred = basel_clf.predict(Xclf_test)

# обосн выбор метр
acur = accuracy_score(yclf_test, yclf_pred)
f_1 = f1_score(yclf_test, yclf_pred)

```

г. Выводы

- согласно книге, влиян. выдвин. гипот. не крит.
- след. выдв. гипот. бол. системн.

4. Имплементация алгоритма машинного обучения

а. Самостоятельно имплементировать алгоритмы машинного обучения (для классификации и регрессии)

б. Обучить имплементированные модели (для классификации и регрессии) для

выбранных наборов данных

с. Оценить качество имплементированных моделей (для классификации и регрессии) по выбранным метрикам на выбранных наборах данных

d. Сравнить результаты имплементированных моделей в сравнении с результатами из пункта 2

е. Сделать выводы

```
In [ ]: from math import sqrt
        from collections import Counter
```

```
In [ ]: # f. Добавить техники из улучшенного бейзлайна (пункт 3с)
        # внедр. в init, fit, predict
        # по книг, n_neighb = 1, допустим. и 3, и 5.

class Custom_Knn_Clf:
    def __init__(self, n_neighbors=1, metric="euclidean",
                 num_cols=None, cat_cols=None):
        self.k = n_neighbors
        self.metric = metric
        self.num_cols = num_cols
        self.cat_cols = cat_cols
        self.scaler = StandardScaler()
        self.encoder = OneHotEncoder(handle_unknown="ignore")
        self.prep_fitted = False

    # в моделях scikit-learn fit возвращает self.
    def fit(self, X, y):
        if self.num_cols or self.cat_cols:
            transformers = []
            if self.num_cols:
                transformers.append(("num", StandardScaler(), self.num_cols))
            if self.cat_cols:
                transformers.append(("cat", OneHotEncoder(handle_unknown="ignore"), self.cat_cols))
            self.prep = ColumnTransformer(transformers)
            self.X_train = self.prep.fit_transform(X)
            self.prep_fitted = True
        else:
            self.X_train = X.values if hasattr(X, "values") else np.array(X)

        self.y_train = np.array(y)
        return self

    # ф-ция metr, частич. замен. упрощ. sklearn

    def metr(self, a, b):
        if self.metric == "euclidean":
            return math.sqrt(sum((ai - bi) ** 2 for ai, bi in zip(a, b)))
```

```

elif self.metric == "manhattan":
    return sum(abs(ai - bi) for ai, bi in zip(a, b))
else:
    raise ValueError("Unsupported metric")

def predict(self, X):
    if self.prep_fitted:
        X_proc = self.prep.transform(X)
    else:
        X_proc = X.values if hasattr(X, "values") else np.array(X)

    predictions = []
    for x in X_proc:
        distances = [self.metr(x, train_x) for train_x in self.X_train]
        k_idx = np.argsort(distances)[:self.k]
        k_labels = [self.y_train[i] for i in k_idx]
        most_common = Counter(k_labels).most_common(1)[0][0]
        predictions.append(most_common)
    return predictions

```

In []: *## g. Обучить модели (для классификации и регрессии) для выбранных наборов*

```

# улучш. clf. препроц. с k=1 и манхет. метр.
prep_clf = Pipeline([
    ("prep", prep_clf_imp),
    ("model", Custom_Knn_Clf(n_neighbors=1, metric="euclidean"))
])

prep_clf.fit(Xclf_train, yclf_train)
yclf_pred_my = prep_clf.predict(Xclf_test)

acc_my = accuracy_score(yclf_test, yclf_pred_my)
f1_my = f1_score(yclf_test, yclf_pred_my)

print(acc_my, f1_my)

# улучш. регр. препроц. с k=1 и манхет. метр.
prep_reg = Pipeline([
    ("prep", preprocessor_reg_imp),
    ("model", Custom_Knn_Reg(n_neighbors=1, metric="manhattan"))
])
# обуч. и предсказ.

prep_reg.fit(Xreg_train, yreg_train)
yreg_pred_my = prep_reg.predict(Xreg_test)
# исп. метр. класса
cust_mae = mean_absolute_error(yreg_train, yreg_pred_my)
cust_rmse = np.sqrt(mean_squared_error(yreg_train, yreg_pred_my))
cust_r2 = r2_score(yreg_train, yreg_pred_my)

print(cust_mae, cust_rmse, cust_r2)

```

In []: *## h. Оценить качество моделей (для классификации и регрессии) по выбранным*

```

print("clf")
print("model---Accuracy---f1")
print("basel {acc:.4f}{f1:.4f}")
print("impr_metr{acc_imp:.4f}{f1_imp:.4f}")
print("knn{acc_my:.4f}{f1_my:.4f}")

```

```
In [ ]: # i. сравнить результаты моделей в сравнении с результатами из пункта 3.  
print("regr")  
print("mae---rmse---r²")  
print("basel{mae:.2f}{rmse:.2f}{r2:.4f}")  
print("impr_metr\{imp_mae:.2f}{imp_rmse:.2f}{imp_r2:.4f}")  
print("custom_metr{cust_mae:.2f}{cust_rmse:.2f}{cust_r2:.4f}")
```

j. сделать выводы

- Улучш. basel. неплох.
- Предобраб. дан. знач. в работе.

```
In [ ]: # han-b set kern via .ven , frmrk  
        # mglearn отстранен, тк без генер данных  
        !pip install pandas numpy matplotlib seaborn scikit-learn
```

Requirement already satisfied: pandas in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (2.3.3)

Requirement already satisfied: numpy in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (2.2.6)

Requirement already satisfied: matplotlib in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (3.10.8)

Requirement already satisfied: mglearn in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (0.2.0)

Requirement already satisfied: seaborn in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (0.13.2)

Requirement already satisfied: scikit-learn in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (1.7.2)

Requirement already satisfied: python-dateutil>=2.8.2 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from pandas) (2.9.0.post0)

Requirement already satisfied: pytz>=2020.1 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from pandas) (2025.2)

Requirement already satisfied: tzdata>=2022.7 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from pandas) (2025.3)

Requirement already satisfied: contourpy>=1.0.1 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (1.3.2)

Requirement already satisfied: cycler>=0.10 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (0.12.1)

Requirement already satisfied: fonttools>=4.22.0 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (4.61.1)

Requirement already satisfied: kiwisolver>=1.3.1 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (1.4.9)

Requirement already satisfied: packaging>=20.0 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (25.0)

Requirement already satisfied: pillow>=8 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (12.0.0)

Requirement already satisfied: pyparsing>=3 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from matplotlib) (3.2.5)

Requirement already satisfied: imageio in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from mglearn) (2.37.2)

Requirement already satisfied: joblib in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from mglearn) (1.5.2)

Requirement already satisfied: scipy>=1.8.0 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from scikit-learn) (1.15.3)

Requirement already satisfied: threadpoolctl>=3.1.0 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from scikit-learn) (3.6.0)

Requirement already satisfied: six>=1.5 in /home/snowwy/Documents/poor_monk/poor_monk/7th_term/ai_course/.venv/lib/python3.10/site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)

1. Выбор начальных условий

а. Набор данных для задачи классификации

Реальная практическая задача:

Выявл. тенденций в мед. страховании между сборами и мед. факторами согласн kaggle-page.

- **Набор данных:**

<https://www.kaggle.com/datasets/muhammadanwaar101/healthcare-insurance-charges-dataset>

б. Набор данных для задачи регрессии

Реальная практическая задача:

Выявл. ХБП и оценки факторов риска функции поч. согласн kaggle-page.

- **Набор данных:** <https://www.kaggle.com/datasets/miadul/kidney-function-health-dataset>

с. Метрики качества и обоснование выбора

- отсылки в коммент. в коде на стр. myuller_gvido

Классификация:

- **Accuracy** - процент объектов, верно классифицированных моделью.
- **F1-score** - среднее гармоническое между точностью (precision) и полнотой (recall).

Регрессия:

- **MAE** - насколько в среднем модель ошибается в прогнозах.
- **R²** - доля дисперсии целевой переменной, которую объясняет модель. Чем ближе к 1, тем лучше.

```
In [10]: # import to get framewrk's functional.  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt
```

```
In [ ]: # 1a. upl classif datst  
# real task - factors influencing insurance charges  
clf_csv_pth = "dat_reg.csv"  
df_clf = pd.read_csv(clf_csv_pth)
```

```
In [ ]: # 1b. upl regr dat
# real task - records w biomarkers
reg_csv_pth = "dat_cl.csv"
df_reg = pd.read_csv(reg_csv_pth)
```

```
In [13]: # observ datafrs

df_clf.head()
# вывод кортеж размерн datafr
df_clf_pd = pd.DataFrame(df_clf)
display(df_clf_pd)

# # первые строк. datafr.
df_reg.head()
df_clf.shape
# вывод обобщ. данн.
df_reg.info()

# IPython.display позволяет "красиво напечатать" датафр
df_reg_pd = pd.DataFrame(df_reg)
display(df_reg_pd)

## Выбрать все строки, в которых значение столбца age больше 30
display(df_reg_pd[df_reg_pd.AGE > 20])
```

	Creatinine	BUN	GFR	Urine_Output	Diabetes	Hypertension	Age
1501	1.157595	12.883540	93.101817	1923.517271	0	0	44.0997
2586	4.795674	63.939097	32.567352	1073.853112	0	0	46.3643
2653	0.793552	18.547813	94.980260	1611.044371	0	1	28.0683
1055	0.916492	17.488186	90.292647	2146.873184	0	0	65.7616
705	1.001983	16.852746	84.319003	1757.950427	1	0	67.0666
...
1987	0.722519	7.944787	99.655555	2412.441089	0	0	44.9672
3648	0.615520	12.631735	90.221509	2218.831898	0	1	71.7788
344	0.910630	12.394649	93.024506	1526.252149	0	1	39.4233
4482	3.120306	60.703081	45.303095	1171.342076	0	1	39.7559
986	6.307465	38.855448	18.040520	971.759106	0	0	55.0813

3000 rows × 11 columns



```
<class 'pandas.core.frame.DataFrame'>
```

```
Index: 844 entries, 353 to 193
```

```
Data columns (total 7 columns):
```

#	Column	Non-Null Count	Dtype
0	AGE	844 non-null	int64
1	Gender	844 non-null	object
2	Body_Mass_Index(BMI)	844 non-null	float64
3	Number_of_Children	844 non-null	int64
4	Smoking_Status	844 non-null	object
5	Region	844 non-null	object
6	Insurance_Charges	844 non-null	float64

```
dtypes: float64(2), int64(2), object(3)
```

```
memory usage: 52.8+ KB
```

	AGE	Gender	Body_Mass_Index(BMI)	Number_of_Children	Smoking_Status	Re
353	33	male	35.245	0	no	north
864	51	male	25.400	0	no	south
1161	34	male	42.130	2	no	south
192	25	male	25.740	0	no	south
782	51	male	35.970	1	no	south
...
421	61	male	35.860	0	yes	south
164	37	male	29.640	0	no	north
28	23	male	17.385	1	no	north
516	20	male	35.310	1	no	south
193	56	female	26.600	1	no	north

844 rows × 7 columns

	AGE	Gender	Body_Mass_Index(BMI)	Number_of_Children	Smoking_Status	Re
353	33	male	35.245	0	no	north
864	51	male	25.400	0	no	south
1161	34	male	42.130	2	no	south
192	25	male	25.740	0	no	south
782	51	male	35.970	1	no	south
...
338	50	male	32.300	1	yes	north
421	61	male	35.860	0	yes	south
164	37	male	29.640	0	no	north
28	23	male	17.385	1	no	north
193	56	female	26.600	1	no	north

738 rows × 7 columns

```
In [ ]: from sklearn.model_selection import train_test_split

# для классиф. раздел. фич на катег (str_cols) и числ (num_cols).
str_cols = df_clf.select_dtypes(include=['object']).columns.tolist()

print("Cat var clf:")
print(str_cols)

str_cols_2 = df_reg.select_dtypes(include=['object']).columns.tolist()
print("Cat var reg:")
print(str_cols_2)
```

Categorical variables:

['Medication']

Categorical variables:

['Gender', 'Smoking_Status', 'Region']

2. Создание бейзлайна и оценка качества

а. Обучить модели из sklearn (для классификации и регрессии) для выбранных наборов данных

```
In [ ]: ## 2a.1 Подгот наб дан.
      ### 2a.1.1 раздел. датафреймовы фич на числ и катег

import matplotlib.pyplot as plt

# необх. отдел. Categorical variables: ['Medication'] и Categorical varia
# перспект.: бин.классиф. при отдел. 'Medication' и 3-ча регр.. при отдел

num_cols_clf = df_clf.drop(columns=["Medication"]).select_dtypes(include=
cat_cols_clf = df_clf.drop(columns=["Medication"]).select_dtypes(include=

num_cols_reg = df_reg.drop(columns=["Medication"]).select_dtypes(include=
cat_cols_reg = df_reg.drop(columns=["Medication"]).select_dtypes(include=
```

```
In [ ]: ### 2a.1.2 опред фич и тарг
X_clf = df_clf.drop(columns=["Medication"])
y_clf = df_clf["Medication"]

X_reg = df_reg.drop(columns=["Medication"])
y_reg = df_reg["Medication"]
```

```
In [ ]: ### 2a.1.3 разб данн согл реком книг myuller_gvido, стр. 32. rand state 0

Xclf_train, Xclf_test, yclf_train, yclf_test = train_test_split(
    X_clf, y_clf, test_size=0.25, random_state=0
)

Xreg_train, Xreg_test, yreg_train, yreg_test = train_test_split(
```

```
X_reg, y_reg, test_size=0.25, random_state=0  
)
```

```
In [ ]: # learnin. basel. лр-3  
  
# clf model, k=3 from myuller_gvido, c.
```

```
In [ ]: from sklearn.preprocessing import StandardScaler, OneHotEncoder  
from sklearn.compose import ColumnTransformer  
from sklearn.neighbors import KNeighborsRegressor  
from sklearn.pipeline import Pipeline  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_s  
from category_encoders import TargetEncoder  
from sklearn.model_selection import GridSearchCV  
from collections import Counter  
from sklearn.metrics import ConfusionMatrixDisplay
```

б. Оценить качество моделей (для классификации и регрессии) по выбранным метрикам на выбранных наборах данных

```
In [ ]: # 2.b. Оценить качество моделей (для классификации и регрессии) по выбран  
print("Classif metr: ", acur, f_1)  
print("Regr metr: ", mae, rmse, r2)  
print("Прогнозы для тестового набора:\n {}".format(y_pred))
```

3. Улучшение бейзлайна

а. Сформулировать гипотезы (препроцессинг данных, визуализация данных, формирование новых признаков, подбор гиперпараметров на кросс-валидации и т.д.)

б. Проверить гипотезы

с. Сформировать улучшенный бейзлайн по результатам проверки гипотез

д. Обучить модели с улучшенным бейзлайном (для классификации и регрессии) для выбранных наборов данных

д. Обучить модели с улучшенным бейзлайном (для классификации и регрессии) для выбранных наборов данных

е. Оценить качество моделей с улучшенным бейзлайном (для классификации и регрессии) по выбранным метрикам на выбранных наборах данных

ф. Сравнить результаты моделей с улучшенным бейзлайном в сравнении с результатами из пункта 2

г. Сделать выводы

```
In [ ]: # гипот.:  
  
# 1. в завис. от rand state меняется кач. train_test_split  
# 2. в завис. от обработ. числ. и категор. столбц. помен. выбр. метрик.  
  
In [ ]: # улучш бейзл  
### 2а.2 Обуч мод  
  
In [ ]: # дл. гипот. 2:  
  
In [ ]: # для clf потенциально масшт. примен. прямое кодирование (при усл. дамми-  
# дл. гипот. 3  
  
In [ ]: # е. оценк. кач-ва модел.  
mae = mean_absolute_error(yreg_test, yreg_pred)  
rmse = np.sqrt(mean_squared_error(yreg_test, yreg_pred))  
r2 = r2_score(yreg_test, yreg_pred)  
# чтоб сдел. прогн., выз. predict объекта  
yclf_pred = basel_clf.predict(Xclf_test)  
  
# обосн выбор метр  
acur = accuracy_score(yclf_test, yclf_pred)  
f_1 = f1_score(yclf_test, yclf_pred)
```

г. Выводы

- согласно книге, влиян. выдвин. гипот. не крит.
- след. выдв. гипот. бол. системн.

4. Имплементация алгоритма машинного обучения

а. Самостоятельно имплементировать алгоритмы машинного обучения (для классификации и регрессии)

б. Обучить имплементированные модели (для классификации и регрессии) для выбранных наборов данных

с. Оценить качество имплементированных моделей (для классификации и регрессии) по выбранным метрикам на выбранных наборах данных

д. Сравнить результаты имплементированных моделей в сравнении с результатами из пункта 2

е. Сделать выводы

```
In [ ]: from math import sqrt
        from collections import Counter
```

```
In [ ]: # f. Добавить техники из улучшенного бейзлайна (пункт 3с)
        # внедр. в init, fit, predict

        # в моделях scikit-learn fit возвращает self.

        # ф-ция metr, частич. замен. упрощ. sklearn

        def metr(self, a, b):
            if self.metric == "euclidean":
                return math.sqrt(sum((ai - bi) ** 2 for ai, bi in zip(a, b)))
            elif self.metric == "manhattan":
                return sum(abs(ai - bi) for ai, bi in zip(a, b))
            else:
                raise ValueError("Unsupported metric")

        def predict(self, X):
            if self.prep_fitted:
                X_proc = self.prep.transform(X)
            else:
                X_proc = X.values if hasattr(X, "values") else np.array(X)

            predictions = []
            for x in X_proc:
                distances = [self.metr(x, train_x) for train_x in self.X_train]
                k_idx = np.argsort(distances)[:self.k]
                k_labels = [self.y_train[i] for i in k_idx]
                most_common = Counter(k_labels).most_common(1)[0][0]
                predictions.append(most_common)
            return predictions
```

```
In [ ]: ## g. Обучить модели (для классификации и регрессии) для выбранных наборов данных.  
  
# обуч. и предсказ.  
# исп. метр. класса
```

```
In [ ]: # h. Оценить качество моделей (для классификации и регрессии) по выбранным метрикам.
```

```
In [ ]: # i. сравнить результаты моделей в сравнении с результатами из пункта 3.
```

j. сделать выводы

- Улучш. basel. неплох.
- Предобrab. дан. знач. в работе.