


▼ **Линейные модели, SVM и деревья решений.**

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 одну из линейных моделей, SVM и 3 дерево решений. Оцените качество моделей с помощью трех подходящих для задачи метрик. Сравните качество полученных моделей.
- 5. Произведите для каждой модели подбор одного гиперпараметра с использованием GridSearchCV и кросс-валидации.
- 6. Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните качество полученных моделей с качеством моделей, полученных в пункте 4.

```
1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6 sns.set(style="ticks")
7 data = pd.read_csv('Data/lab_5/winequalityN.csv', sep=",")
8 data.head(5)
```




	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.01
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.99
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.99
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.99

```
1 data.shape
```



(6497, 13)

```
1 # Кодирование категориального признака(тип вина: красное или белое) в столбец wine_type_le
2 from sklearn.preprocessing import LabelEncoder
3 le = LabelEncoder()
4 le.fit(data.type)
5 data['wine_type_le'] = le.transform(data.type)
6 data.head(2)
```



	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.01
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.99

```
1 del data['type']
```

```
1 data.head(2)
```



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.001
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.994

```
1 # Проверка на пустые значения
2 data.isnull().sum()
```



```
fixed acidity          10
volatile acidity       8
citric acid            3
residual sugar         2
chlorides              2
free sulfur dioxide    0
total sulfur dioxide   0
density               0
pH                    9
sulphates              4
alcohol               0
quality               0
wine_type_le          0
dtype: int64
```

```
1 import pandas as pd
2 # function to clean the dataset of nan, Inf, and missing cells (for skewed data)
3 def clean_dataset(df):
4     assert isinstance(df, pd.DataFrame), "df needs to be a pd.DataFrame"
5     df.dropna(inplace=True)
6     indices_to_keep = ~df.isin([np.nan, np.inf, -np.inf]).any(1)
7     return df[indices_to_keep].astype(np.float64)
```

```
1 clean_dataset(data)[:1]
```




	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.001

```
1 # Пустых значений нет
2 # Перейдем к разделению выборки на обучающую и тестовую.
3 X = data.drop('wine_type_le',axis = 1).values
4 y = data['wine_type_le'].values
```


```
1 from sklearn.model_selection import train_test_split
2 # Функция train_test_split разделила исходную выборку таким образом,
3 #чтобы в обучающей и тестовой частях сохранились пропорции классов.
4 X_train, X_test, y_train, y_test = train_test_split(
5     X, y, test_size=0.30, random_state=1)
```

```
1 # Размер обучающей выборки (70%)
```


```
2 print('X_train: {} y_train: {}'.format(X_train.shape, y_train.shape))
```

 X\_train: (4524, 12) y\_train: (4524,)


```
1 # Размер тестовой выборки (30%)
2 print('X_test: {} y_test: {}'.format(X_test.shape, y_test.shape))
```

 X\_test: (1939, 12) y\_test: (1939,)

```
1 # Функция train_test_split разделила исходную выборку таким образом,
2 # чтобы в обучающей и тестовой частях сохранились пропорции классов.
3 np.unique(y_train)
```

 array([0, 1])

```
1 np.unique(y_test)
```

 array([0, 1])

```
1 from sklearn.linear_model import SGDClassifier
2 from sklearn.svm import LinearSVC
3 from sklearn.tree import DecisionTreeClassifier
4 from sklearn.model_selection import GridSearchCV
5
6 from sklearn.metrics import accuracy_score
7 from sklearn.metrics import balanced_accuracy_score
8 from sklearn.metrics import precision_score, recall_score, f1_score
```

## ▼ Сравнение качества трех линейных моделей

### ► SGDClassifier (градиентный метод)

↳ 5 cells hidden

### ► LinearSVC (линейный)

↳ 5 cells hidden

### ► DecisionTreeClassifier (дерево решений)

↳ 6 cells hidden

### ► Подбор одного гиперпараметра с использованием GridSearchCV и кросс-валидации

↳ 12 cells hidden

### ► Сравнение качества полученных моделей с качеством моделей, полученных ранее

## SGD

↳ 5 cells hidden

### ▶ LinearSVC

↳ 5 cells hidden

### ▶ DecisionTree

↳ 6 cells hidden