**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №4

Выполнил:  
Студент группы ИУ5-62Б   
Андреев Илья

Подпись и дата:

Проверил:  
преподаватель каф. ИУ5   
Гапанюк Ю.Е.

Подпись и дата:

г. Москва, 2020 г.

# Лабораторная работа 4

## Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей.

**Цель лабораторной работы:** изучение сложных способов подготовки выборки и подбора гиперпараметров на примере метода ближайших соседей.

### Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. С использованием метода train\_test\_split разделите выборку на обучающую и тестовую.
3. Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью подходящих для задачи метрик.
4. Постройте модель и оцените качество модели с использованием кросс-валидации.
5. Произведите подбор гиперпараметра K с использованием GridSearchCV и кросс-валидации.

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **seaborn** **as** **sns**

**import** **matplotlib.pyplot** **as** **plt**

%**matplotlib** inline

**from** **sklearn.impute** **import** SimpleImputer

**import** **pandas\_profiling** **as** **pp**

**import** **warnings**

warnings.simplefilter("ignore")

**Загрузка и первичный анализ**

In [7]:

data = pd.read\_csv('train.csv')

In [8]:

data.head()

Out[8]:

|  | **PassengerId** | **Survived** | **Pclass** | **Name** | **Sex** | **Age** | **SibSp** | **Parch** | **Ticket** | **Fare** | **Cabin** | **Embarked** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |

In [9]:

pp.ProfileReport(data)

Out[9]:

**Overview**

Dataset info

|  |  |
| --- | --- |
| **Number of variables** | 12 |
| **Number of observations** | 891 |
| **Total Missing (%)** | 8.1% |
| **Total size in memory** | 83.7 KiB |
| **Average record size in memory** | 96.1 B |

Variables types

|  |  |
| --- | --- |
| **Numeric** | 6 |
| **Categorical** | 4 |
| **Boolean** | 1 |
| **Date** | 0 |
| **Text (Unique)** | 1 |
| **Rejected** | 0 |
| **Unsupported** | 0 |

Warnings

* [Age](http://localhost:8888/nbconvert/html/Desktop/Lab4.ipynb?download=false#pp_var_Age) has 177 / 19.9% missing values **Missing**
* [SibSp](http://localhost:8888/nbconvert/html/Desktop/Lab4.ipynb?download=false#pp_var_SibSp) has 608 / 68.2% zeros **Zeros**
* [Parch](http://localhost:8888/nbconvert/html/Desktop/Lab4.ipynb?download=false#pp_var_Parch) has 678 / 76.1% zeros **Zeros**
* [Ticket](http://localhost:8888/nbconvert/html/Desktop/Lab4.ipynb?download=false#pp_var_Ticket) has a high cardinality: 681 distinct values **Warning**
* [Fare](http://localhost:8888/nbconvert/html/Desktop/Lab4.ipynb?download=false#pp_var_Fare) has 15 / 1.7% zeros **Zeros**
* [Cabin](http://localhost:8888/nbconvert/html/Desktop/Lab4.ipynb?download=false#pp_var_Cabin) has 687 / 77.1% missing values **Missing**
* [Cabin](http://localhost:8888/nbconvert/html/Desktop/Lab4.ipynb?download=false#pp_var_Cabin) has a high cardinality: 148 distinct values **Warning**

**Variables**

PassengerId  
Numeric

|  |  |
| --- | --- |
| **Distinct count** | 891 |
| **Unique (%)** | 100.0% |
| **Missing (%)** | 0.0% |
| **Missing (n)** | 0 |
| **Infinite (%)** | 0.0% |
| **Infinite (n)** | 0 |
| **Mean** | 446 |
| **Minimum** | 1 |
| **Maximum** | 891 |
| **Zeros (%)** | 0.0% |

Toggle details

Survived  
Boolean

|  |  |  |  |
| --- | --- | --- | --- |
| **Distinct count** | | 2 | |
| **Unique (%)** | | 0.2% | |
| **Missing (%)** | | 0.0% | |
| **Missing (n)** | | 0 | |
| **Mean** | 0.38384 |
| 0 | | | 549 | |
| 1 | | | 342 | |

Toggle details

Pclass  
Numeric

|  |  |
| --- | --- |
| **Distinct count** | 3 |
| **Unique (%)** | 0.3% |
| **Missing (%)** | 0.0% |
| **Missing (n)** | 0 |
| **Infinite (%)** | 0.0% |
| **Infinite (n)** | 0 |
| **Mean** | 2.3086 |
| **Minimum** | 1 |
| **Maximum** | 3 |
| **Zeros (%)** | 0.0% |

Toggle details

Name  
Categorical, Unique

| **First 3 values** |
| --- |
| Hale, Mr. Reginald |
| Pavlovic, Mr. Stefo |
| Smith, Miss. Marion Elsie |
| **Last 3 values** |
| Gustafsson, Mr. Karl Gideon |
| Thayer, Mr. John Borland |
| Masselmani, Mrs. Fatima |

Toggle details

Sex  
Categorical

|  |  |
| --- | --- |
| **Distinct count** | 2 |
| **Unique (%)** | 0.2% |
| **Missing (%)** | 0.0% |
| **Missing (n)** | 0 |
| male | | 577 |
| female | | 314 |

Toggle details

Age  
Numeric

|  |  |
| --- | --- |
| **Distinct count** | 89 |
| **Unique (%)** | 10.0% |
| **Missing (%)** | 19.9% |
| **Missing (n)** | 177 |
| **Infinite (%)** | 0.0% |
| **Infinite (n)** | 0 |
| **Mean** | 29.699 |
| **Minimum** | 0.42 |
| **Maximum** | 80 |
| **Zeros (%)** | 0.0% |

Toggle details

SibSp  
Numeric

|  |  |
| --- | --- |
| **Distinct count** | 7 |
| **Unique (%)** | 0.8% |
| **Missing (%)** | 0.0% |
| **Missing (n)** | 0 |
| **Infinite (%)** | 0.0% |
| **Infinite (n)** | 0 |
| **Mean** | 0.52301 |
| **Minimum** | 0 |
| **Maximum** | 8 |
| **Zeros (%)** | 68.2% |

Toggle details

Parch  
Numeric

|  |  |
| --- | --- |
| **Distinct count** | 7 |
| **Unique (%)** | 0.8% |
| **Missing (%)** | 0.0% |
| **Missing (n)** | 0 |
| **Infinite (%)** | 0.0% |
| **Infinite (n)** | 0 |
| **Mean** | 0.38159 |
| **Minimum** | 0 |
| **Maximum** | 6 |
| **Zeros (%)** | 76.1% |

Toggle details

Ticket  
Categorical

|  |  |
| --- | --- |
| **Distinct count** | 681 |
| **Unique (%)** | 76.4% |
| **Missing (%)** | 0.0% |
| **Missing (n)** | 0 |
| 347082 | | 7 |
| CA. 2343 | | 7 |
| 1601 | | 7 |
| Other values (678) | | 870 |

Toggle details

Fare  
Numeric

|  |  |
| --- | --- |
| **Distinct count** | 248 |
| **Unique (%)** | 27.8% |
| **Missing (%)** | 0.0% |
| **Missing (n)** | 0 |
| **Infinite (%)** | 0.0% |
| **Infinite (n)** | 0 |
| **Mean** | 32.204 |
| **Minimum** | 0 |
| **Maximum** | 512.33 |
| **Zeros (%)** | 1.7% |

Toggle details

Cabin  
Categorical

|  |  |
| --- | --- |
| **Distinct count** | 148 |
| **Unique (%)** | 16.6% |
| **Missing (%)** | 77.1% |
| **Missing (n)** | 687 |
| G6 | | 4 |
| B96 B98 | | 4 |
| C23 C25 C27 | | 4 |
| Other values (144) | | 192 |
| (Missing) | | 687 |

Toggle details

Embarked  
Categorical

|  |  |
| --- | --- |
| **Distinct count** | 4 |
| **Unique (%)** | 0.4% |
| **Missing (%)** | 0.2% |
| **Missing (n)** | 2 |
| S | | 644 |
| C | | 168 |
| Q | | 77 |
| (Missing) | | 2 |

Toggle details

**Correlations**

**Sample**

|  | **PassengerId** | **Survived** | **Pclass** | **Name** | **Sex** | **Age** | **SibSp** | **Parch** | **Ticket** | **Fare** | **Cabin** | **Embarked** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Thayer) | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |

In [4]:

data.shape

Out[4]:

(891, 12)

In [5]:

data.dtypes

Out[5]:

PassengerId int64

Survived int64

Pclass int64

Name object

Sex object

Age float64

SibSp int64

Parch int64

Ticket object

Fare float64

Cabin object

Embarked object

dtype: object

In [6]:

data.isnull().sum()

Out[6]:

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 177

SibSp 0

Parch 0

Ticket 0

Fare 0

Cabin 687

Embarked 2

dtype: int64

In [7]:

data.info()

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

RangeIndex: 891 entries, 0 to 890

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 PassengerId 891 non-null int64

1 Survived 891 non-null int64

2 Pclass 891 non-null int64

3 Name 891 non-null object

4 Sex 891 non-null object

5 Age 714 non-null float64

6 SibSp 891 non-null int64

7 Parch 891 non-null int64

8 Ticket 891 non-null object

9 Fare 891 non-null float64

10 Cabin 204 non-null object

11 Embarked 889 non-null object

dtypes: float64(2), int64(5), object(5)

memory usage: 83.7+ KB

**Обработка пропусков**

In [8]:

*# Удаляем столбцы, которые не несут значимой информации*

data.drop(['Cabin','Name','Ticket','PassengerId'], axis = 1, inplace = **True**)

y = data.Survived

data.drop('Survived', axis=1, inplace=**True**)

In [9]:

data.info()

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

RangeIndex: 891 entries, 0 to 890

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Pclass 891 non-null int64

1 Sex 891 non-null object

2 Age 714 non-null float64

3 SibSp 891 non-null int64

4 Parch 891 non-null int64

5 Fare 891 non-null float64

6 Embarked 889 non-null object

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

memory usage: 48.9+ KB

In [10]:

*# Заполняем отсутствующие значения возраста средним возрастом*

data['Age'] = data['Age'].replace(0,np.nan)

data['Age'] = data['Age'].fillna(data['Age'].mean())

In [11]:

data.isnull().sum()

Out[11]:

Pclass 0

Sex 0

Age 0

SibSp 0

Parch 0

Fare 0

Embarked 2

dtype: int64

**Преобразование категориальных признаков в числовые**

In [12]:

data['Embarked'].value\_counts()

Out[12]:

S 644

C 168

Q 77

Name: Embarked, dtype: int64

In [13]:

*# Кодируем признаки Pclass и Embarked в отдельные столбцы*

data = pd.get\_dummies(data, columns=['Pclass','Embarked'])

In [14]:

*# Пол кодируем в 1/0*

data['IsMale']=data.Sex.replace({'female':0,'male':1})

data.drop('Sex', axis = 1, inplace = **True**)

In [15]:

data.head()

Out[15]:

|  | **Age** | **SibSp** | **Parch** | **Fare** | **Pclass\_1** | **Pclass\_2** | **Pclass\_3** | **Embarked\_C** | **Embarked\_Q** | **Embarked\_S** | **IsMale** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 22.0 | 1 | 0 | 7.2500 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| **1** | 38.0 | 1 | 0 | 71.2833 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| **2** | 26.0 | 0 | 0 | 7.9250 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| **3** | 35.0 | 1 | 0 | 53.1000 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| **4** | 35.0 | 0 | 0 | 8.0500 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

**Масштабирование значений**

In [16]:

**from** **sklearn.preprocessing** **import** StandardScaler

In [17]:

sc2 = StandardScaler()

sc2.fit(data)

sc2\_data = sc2.transform(data)

In [18]:

data = pd.DataFrame(sc2\_data, columns = data.columns)

In [19]:

data.head()

Out[19]:

|  | **Age** | **SibSp** | **Parch** | **Fare** | **Pclass\_1** | **Pclass\_2** | **Pclass\_3** | **Embarked\_C** | **Embarked\_Q** | **Embarked\_S** | **IsMale** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | -0.592481 | 0.432793 | -0.473674 | -0.502445 | -0.565685 | -0.510152 | 0.902587 | -0.482043 | -0.307562 | 0.619306 | 0.737695 |
| **1** | 0.638789 | 0.432793 | -0.473674 | 0.786845 | 1.767767 | -0.510152 | -1.107926 | 2.074505 | -0.307562 | -1.614710 | -1.355574 |
| **2** | -0.284663 | -0.474545 | -0.473674 | -0.488854 | -0.565685 | -0.510152 | 0.902587 | -0.482043 | -0.307562 | 0.619306 | -1.355574 |
| **3** | 0.407926 | 0.432793 | -0.473674 | 0.420730 | 1.767767 | -0.510152 | -1.107926 | -0.482043 | -0.307562 | 0.619306 | -1.355574 |
| **4** | 0.407926 | -0.474545 | -0.473674 | -0.486337 | -0.565685 | -0.510152 | 0.902587 | -0.482043 | -0.307562 | 0.619306 | 0.737695 |

**Разделение выборки**

In [20]:

**from** **sklearn.model\_selection** **import** train\_test\_split

In [21]:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data, y, test\_size=0.2)

print (X\_train.shape, y\_train.shape)

print (X\_test.shape, y\_test.shape)

(712, 11) (712,)

(179, 11) (179,)

**Обучение модели**

In [22]:

**from** **sklearn.neighbors** **import** KNeighborsClassifier

In [23]:

KNeighborsClassifierObj = KNeighborsClassifier(n\_neighbors=10)

In [24]:

KNeighborsClassifierObj.fit(X\_train, y\_train)

Out[24]:

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=10, p=2,

weights='uniform')

In [25]:

y\_predicted = KNeighborsClassifierObj.predict(X\_test)

**Метрики качества**

In [26]:

**from** **sklearn.metrics** **import** accuracy\_score, balanced\_accuracy\_score, precision\_score, f1\_score, classification\_report

In [27]:

accuracy\_score(y\_test, y\_predicted)

Out[27]:

0.8100558659217877

In [28]:

balanced\_accuracy\_score(y\_test, y\_predicted)

Out[28]:

0.7855869620575503

In [29]:

precision\_score(y\_test, y\_predicted)

Out[29]:

0.9215686274509803

In [30]:

classification\_report(y\_test, y\_predicted, output\_dict = **True**)

Out[30]:

{'0': {'precision': 0.765625,

'recall': 0.9607843137254902,

'f1-score': 0.8521739130434782,

'support': 102},

'1': {'precision': 0.9215686274509803,

'recall': 0.6103896103896104,

'f1-score': 0.734375,

'support': 77},

'accuracy': 0.8100558659217877,

'macro avg': {'precision': 0.8435968137254901,

'recall': 0.7855869620575503,

'f1-score': 0.7932744565217391,

'support': 179},

'weighted avg': {'precision': 0.8327068956074049,

'recall': 0.8100558659217877,

'f1-score': 0.8015006376001943,

'support': 179}}

**Кросс-валидация**

In [31]:

**from** **sklearn.model\_selection** **import** cross\_val\_score

In [32]:

scores = cross\_val\_score(KNeighborsClassifierObj,

X\_train, y\_train, cv=3,

scoring='f1\_weighted')

scores, np.mean(scores)

Out[32]:

(array([0.74089857, 0.79170005, 0.80583123]), 0.7794766175045528)

**Подбор гиперпараметров**

In [33]:

**from** **sklearn.model\_selection** **import** GridSearchCV

In [34]:

n\_range = np.array(range(5,55,5))

tuned\_parameters = [{'n\_neighbors': n\_range}]

In [35]:

clf\_gs = GridSearchCV(KNeighborsClassifier(), tuned\_parameters, cv=5, scoring='f1\_weighted')

In [36]:

clf\_gs.fit(X\_train, y\_train)

Out[36]:

GridSearchCV(cv=5, error\_score=nan,

estimator=KNeighborsClassifier(algorithm='auto', leaf\_size=30,

metric='minkowski',

metric\_params=None, n\_jobs=None,

n\_neighbors=5, p=2,

weights='uniform'),

iid='deprecated', n\_jobs=None,

param\_grid=[{'n\_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])}],

pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score=False,

scoring='f1\_weighted', verbose=0)

In [37]:

clf\_gs.best\_params\_

Out[37]:

{'n\_neighbors': 20}

In [38]:

clf\_gs.best\_score\_

Out[38]:

0.786847821819429

In [39]:

plt.plot(n\_range, clf\_gs.cv\_results\_['mean\_test\_score'])

Out[39]:

[<matplotlib.lines.Line2D at 0x1b01a86ecc8>]

