

به نام خدا

# پیاده سازی الگوریتم KNN

گردآورنده: ایمان جوادی سیسی

درس: داده کاوی

مقطع: کارشناسی

دانشگاه آزاد اسلامی واحد علوم و تحقیقات

۱۴۰۳ آذر ۱۰

میخواهیم با استفاده از دیتابس titanic\_train الگوریتم KNN را پیاده سازی کنیم.

ابتدا کتابخانه های مورد نظر را import میکنیم :

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split,
    StratifiedShuffleSplit

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import OneHotEncoder

from sklearn.preprocessing import LabelEncoder

from sklearn.pipeline import Pipeline

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix,
    classification_report

from sklearn.model_selection import cross_val_score
```

## ۱. توصیف داده ها:

### ۱.۱ فراخوانی داده ها

با استفاده از دستور `read_csv` فایل csv را باز میکنیم.

```
titanic = pd.read_csv("titanic_train.csv")
```

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
...	...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W/C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

از دستوراتی نظیر `head()`, `tail()` میتوانیم برای دیدن ۵ دیتای اول و ۵ دیتای آخر استفاده کنیم.

```
titanic.head()
```

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

```
titanic.tail()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W/C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

## ۲.۱ اطلاعات داده ها

برای دیدن اطلاعاتی مانند تعداد ستون های دیتاست یا نوع دیتا ، از دستور info() استفاده میکنیم

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Survived    891 non-null    int64  
 1   Pclass       891 non-null    int64  
 2   Sex          891 non-null    int32  
 3   Age          891 non-null    float64 
 4   SibSp        891 non-null    int64  
 5   Parch        891 non-null    int64  
 6   Fare          891 non-null    float64 
 7   Embarked     891 non-null    int32  
 8   Ticket_number 891 non-null    float64 
 9   Ticket_Code   891 non-null    int32  
dtypes: float64(3), int32(3), int64(4)
memory usage: 59.3 KB
```

## ۳.۱ داده های دارای مقادیر خالی

تعداد مقادیر خالی را برای هر ستون محاسبه میکند .

```
titanic.isnull().sum()
```

```
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
```

## ۴.۱ آمار توصیفی داده ها

ارائه آمارهای توصیفی برای داده های عددی.

```
titanic.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

## ۵.۱ اندازه مجموعه داده ها

```
titanic.shape
```

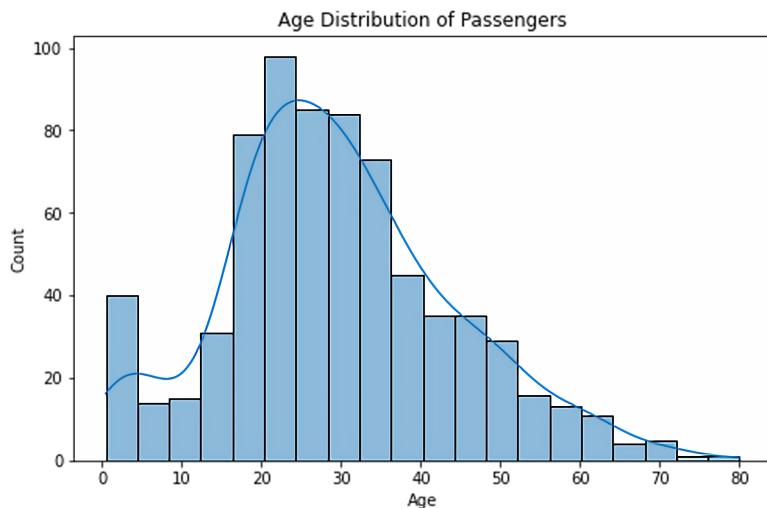
(891, 12)

## ۲. بصری سازی داده ها

میتوان با توجه به نیاز ، انواع نمودار ها را با استفاده از کتابخانه های matplotlib seaborn نمایش گذاشت.

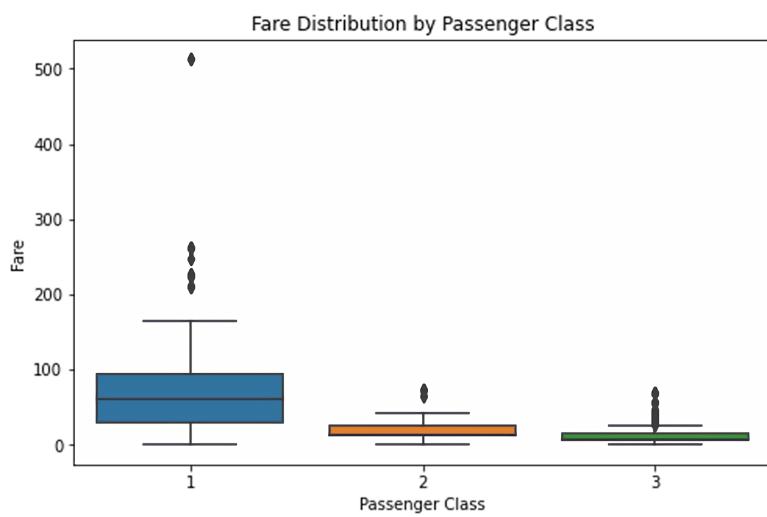
### ۱.۲ توزیع سنی

```
plt.figure(figsize=(8, 5))
sns.histplot(data=titanic, x='Age', bins=20, kde=True)
plt.title('Age Distribution of Passengers')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



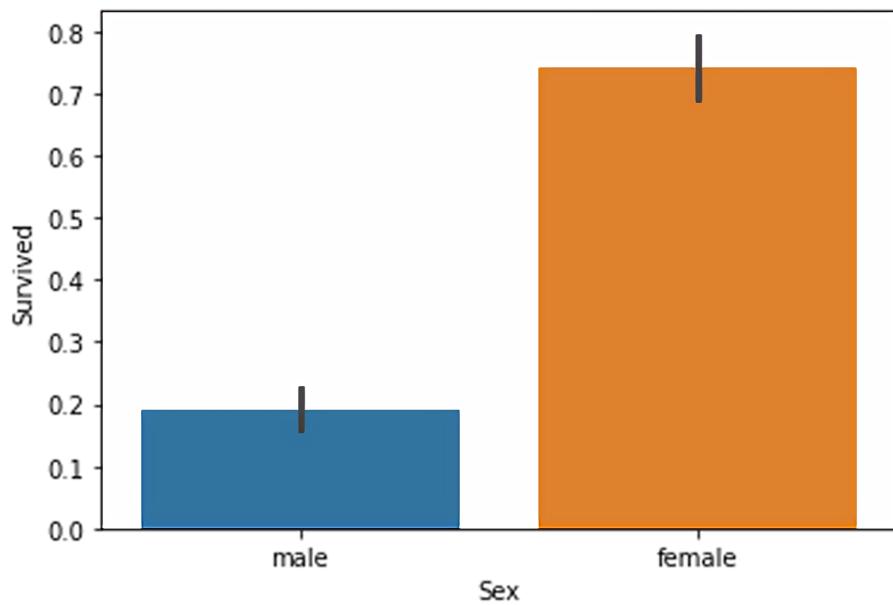
٢.٢ توزيع كلاس cabin

```
plt.figure(figsize=(8, 5))
sns.boxplot(data=titanic, x='Pclass', y='Fare')
plt.title('Fare Distribution by Passenger Class')
plt.xlabel('Passenger Class')
plt.ylabel('Fare')
plt.show()
```



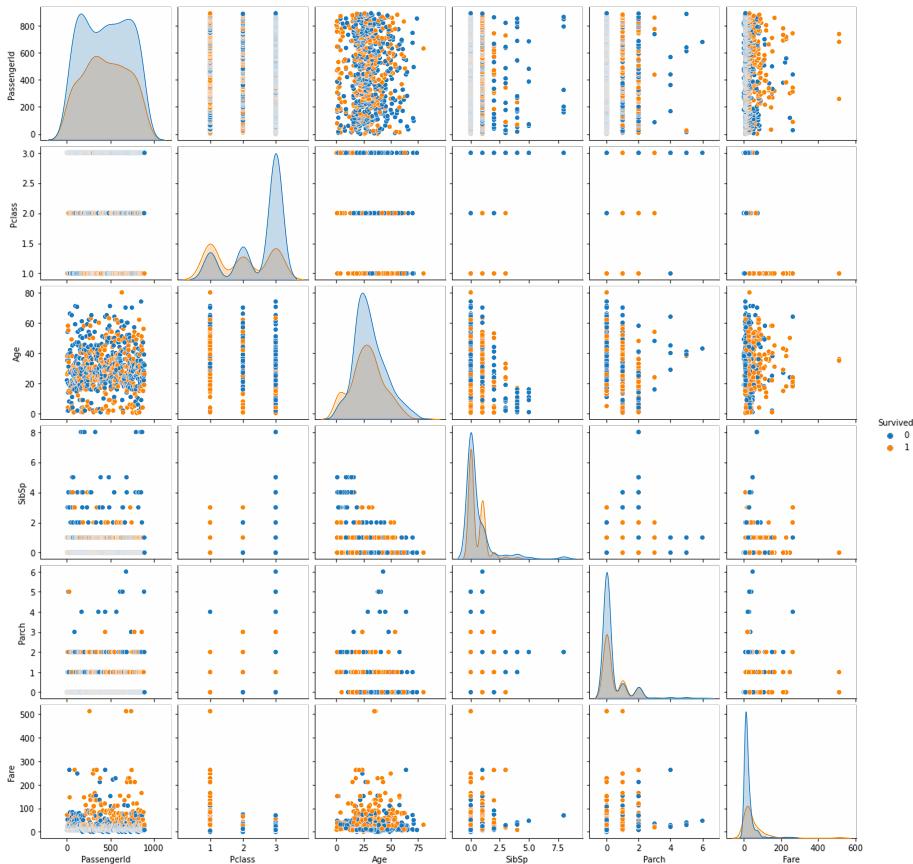
## ٣.٢ توزيع جنسیت

```
sns.barplot(x=titanic['Sex'],y=titanic['Survived'])  
plt.show()
```



## ٤.٢ روابط ستون ها

```
sns.pairplot(titanic, hue='Survived')  
plt.show()
```



### ۳. پیش پردازش داده ها:

۱.۳ پاک کردن ستون های غیر ضروری

```
titanic.drop(columns=['PassengerId' , 'Name' , 'Cabin'] ,  
           inplace=True)
```

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 891 entries, 0 to 890  
Data columns (total 9 columns):  
 #   Column      Non-Null Count  Dtype     
---    
 0   Survived    891 non-null    int64    
 1   Pclass       891 non-null    int64    
 2   Sex          891 non-null    object    
 3   Age          714 non-null    float64   
 4   SibSp        891 non-null    int64    
 5   Parch        891 non-null    int64    
 6   Ticket       891 non-null    object    
 7   Fare          891 non-null    float64   
 8   Embarked     889 non-null    object    
dtypes: float64(2), int64(4), object(3)  
memory usage: 62.8+ KB
```

۲.۳ مقادیر از دست رفته

در ستون Age مقدار Nan داریم. این مقدار Nan را با میانگین پر می کنیم.

```
titanic['Age'].fillna(titanic['Age'].mean() , inplace=True)
```

ما در ستون Embarked مقدار Nan داریم. این مقدار Nan را با رایج ترین عنصر پر می کنیم.

```
titanic['Embarked'] =  
titanic['Embarked'].fillna(titanic['Embarked'].mode()[0])
```

## ۴. پردازش متن و ویژگی‌های دسته‌بندی شده:

۱.۶ کنترل کردن ویژگی‌های جنسیت و مکان سوار شدن:

```
le = LabelEncoder()  
cats_col = ['Sex', 'Embarked']  
  
for cat in cats_col:  
    titanic[cat] = le.fit_transform(titanic[cat])
```

```
titanic.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	0	3	1	22.0	1	0	A/5 21171	7.2500	2
1	1	1	0	38.0	1	0	PC 17599	71.2833	0
2	1	3	0	26.0	0	0	STON/O2. 3101282	7.9250	2
3	1	1	0	35.0	1	0	113803	53.1000	2
4	0	3	1	35.0	0	0	373450	8.0500	2

## ۲.۶ کنترل کردن ویژگی بلیط

جداسازی داده‌های عددی و غیرعددی در ستون ticket و ایجاد دو ستون جدید Ticket\_number و ایجاد دو ستون جدید Ticket\_code برای قرار دادن مقادیر.

```
def preprocess(df):  
  
    def normalize_name(x):  
        return " ".join([v.strip(",()[]\\"") for v in x.split(" ")])  
  
    def ticket_number(x):  
        return x.split(" ")[-1]  
  
    def ticket_item(x):
```

```

        items = x.split(" ")
        if len(items) == 1:
            return "NONE"
        return "_" .join(items[0:-1])

df["Ticket_number"] = df["Ticket"].apply(ticket_number)
df["Ticket_Code"] = df["Ticket"].apply(ticket_item)

df = df.drop(['Ticket'], axis=1)

return df

titanic = preprocess(titanic)

titanic.head()

```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Ticket_number	Ticket_Code
0	0	3	1	22.0	1	0	7.2500	2	21171	A/5
1	1	1	0	38.0	1	0	71.2833	0	17599	PC
2	1	3	0	26.0	0	0	7.9250	2	3101282	STON/O2.
3	1	1	0	35.0	1	0	53.1000	2	113803	NONE
4	0	3	1	35.0	0	0	8.0500	2	373450	NONE

```
titanic.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Survived    891 non-null    int64  
 1   Pclass      891 non-null    int64  
 2   Sex         891 non-null    int32  
 3   Age         891 non-null    float64 
 4   SibSp       891 non-null    int64  
 5   Parch       891 non-null    int64  
 6   Fare         891 non-null    float64 
 7   Embarked    891 non-null    int32  
 8   Ticket_number 891 non-null    object  
 9   Ticket_Code  891 non-null    object  
dtypes: float64(2), int32(2), int64(4), object(2)
memory usage: 62.8+ KB

```

تبدیل داده های ستون Ticket\_code به نوع داده عددی int

```
titanic['Ticket_Code'] = le.fit_transform(titanic['Ticket_Code'])
```

تبدیل داده های ستون Ticket\_number به نوع داده عددی float

```
titanic['Ticket_number'] = pd.to_numeric(titanic['Ticket_number'],
                                         errors='coerce')
```

در کد زیر اعداد تصادفی بین میانگین و انحراف استاندارد مقادیر غیر-NaN را تولید میکنیم و سپس مقادیر NaN با اعداد تصادفی تولید شده پر میکنیم تا بتوانیم دقت مدل را بالا تر ببریم.

```
columns_to_fill = ['Ticket_number']

for column in columns_to_fill:

    mean_value = titanic[column].mean()
    std_value = titanic[column].std()
    random_values = np.random.normal(mean_value, std_value,
                                      size=titanic[column].
                                      isnull().sum())

    titanic.loc[titanic[column].isnull(), column] =
    random_values
```

```
titanic.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Survived    891 non-null    int64  
 1   Pclass       891 non-null    int64  
 2   Sex          891 non-null    int32  
 3   Age          891 non-null    float64 
 4   SibSp        891 non-null    int64  
 5   Parch        891 non-null    int64  
 6   Fare          891 non-null    float64 
 7   Embarked     891 non-null    int32  
 8   Ticket number 891 non-null    float64 
 9   Ticket_Code  891 non-null    int32  
dtypes: float64(3), int32(3), int64(4)
memory usage: 59.3 KB

```

## ۵. ایجاد یک مجموعه تست با نمونه‌گیری طبقه‌بندی شده:

فرض کنید مجموعه داده `titanic` یک Data Frame است که هم ویژگی‌ها و هم متغیر هدف را در خود جای داده است. در این سری کد ما تلاش می‌کنیم تا ویژگی‌های  $X$  و متغیر هدف  $Y$  را جدا کنیم.

```

X = titanic.drop('Survived', axis=1)
y = titanic['Survived']

```

```

split = StratifiedShuffleSplit(n_splits=1, test_size=0.2,
                               random_state=42)

```

```

for train_index, test_index in split.split(X, y):
    strat_train_set = titanic.loc[train_index]
    strat_test_set = titanic.loc[test_index]

```

```

X_train = X.loc[train_index]
y_train = y.loc[train_index]
X_test = X.loc[test_index]
y_test = y.loc[test_index]

```

## ۶. مجموعه‌ای از تبدیل‌ها:

```
num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('std_scaler', StandardScaler()),
])

num_attribs = list(titanic)
cat_attribs = ['Sex', 'Embarked']

full_pipeline = ColumnTransformer([
    ('num', num_pipeline, num_attribs),
    ('cat', OneHotEncoder(), cat_attribs),
])

titanic_prepared = full_pipeline.fit_transform(strat_train_set)
```

## ۷. انتخاب و آموزش مدل:

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(titanic_prepared, strat_train_set['Survived'])
```

```
KNeighborsClassifier(n_neighbors=3)
```

۸. پیش‌بینی‌ها و ارزیابی‌ها شامل ماتریس درهم‌ریختگی هستند:

```
titanic_test_prepared = full_pipeline.transform(strat_test_set)
predictions = knn.predict(titanic_test_prepared)
conf_matrix = confusion_matrix(strat_test_set['Survived'],
                               predictions)
```

```
predictions
```

```
array([0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0], dtype=int64)
```

```
conf_matrix
```

```
array([[109,    1],
       [  2,  67]], dtype=int64)
```

```
classification_report(strat_test_set['Survived'], predictions,
                      output_dict=True)
```

```
{'0': {'precision': 0.9819819819819819,
'recall': 0.990909090909091,
'f1-score': 0.9864253393665158,
'support': 110},
'1': {'precision': 0.9852941176470589,
'recall': 0.9710144927536232,
'f1-score': 0.9781021897810219,
'support': 69},
'accuracy': 0.9832402234636871,
'macro avg': {'precision': 0.9836380498145204,
'recall': 0.9809617918313571,
'f1-score': 0.9822637645737688,
'support': 179},
'weighted avg': {'precision': 0.9832587270148886,
'recall': 0.9832402234636871,
'f1-score': 0.9832169744424986,
'support': 179}}
```

```
best_k = 1
best_score = 0

for k in range(1, 21):
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, titanic_prepared,
                             strat_train_set['Survived'], cv=10)
    mean_score = scores.mean()
    if mean_score > best_score:
        best_k = k
        best_score = mean_score
print("Best k:", best_k)
print("Best Cross-Validation Score:", best_score)
```

```
Best k: 1
Best Cross-Validation Score: 0.9901799687010954
```