

## 1. Импорт данных

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import cross_val_score
import sys
import io

sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')
```

## 2. Загрузка данных и выделение обучающей и тестовой выборки

```
# Загрузка обучающей и тестовой выборок
column_names = [
    'age', 'workclass', 'fnlwgt', 'education', 'education-num',
    'marital-status', 'occupation', 'relationship', 'race', 'sex',
    'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', 'income'
]

# Загрузка данных
train_data = pd.read_csv(r'D:\VSU\sem7\AI\lab3\adult.data', names=column_names,
sep=',\s*', na_values='?', engine='python')
test_data = pd.read_csv(r'D:\VSU\sem7\AI\lab3\adult.test', names=column_names,
sep=',\s*', na_values='?', skiprows=1, engine='python')

# Предобработка целевой переменной (удаление точки в тестовой выборке)
test_data['income'] = test_data['income'].str.replace('.', '')

# Объединение данных для удобства предобработки
data = pd.concat([train_data, test_data], ignore_index=True)

# Заполнение пропущенных значений
for column in data.columns:
    if data[column].dtype == 'object':
        data[column].fillna(data[column].mode()[0], inplace=True)
    else:
        data[column].fillna(data[column].median(), inplace=True)

# Кодирование категориальных переменных
label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    if column != 'income':
        le = LabelEncoder()
        data[column] = le.fit_transform(data[column])
        label_encoders[column] = le
```

```

# Кодирование целевой переменной
income_encoder = LabelEncoder()
data['income'] = income_encoder.fit_transform(data['income'])

# Разделение обратно на обучающую и тестовую выборки
X_train = data.iloc[:len(train_data)].drop('income', axis=1)
y_train = data.iloc[:len(train_data)]['income']
X_test = data.iloc[len(train_data):].drop('income', axis=1)
y_test = data.iloc[len(train_data):]['income']

print(f"Обучающая выборка: {X_train.shape[0]} записей")
print(f"Тестовая выборка: {X_test.shape[0]} записей")

```

### 3. Определяем наилучшее значение k

```

print("\n2. Определение наилучшего значения k")

best_k = None
best_accuracy = 0

for k in range(1, 11): # Перебираем значения k от 1 до 10
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_train, y_train, cv=5) # 5-кратная
    кроссквалидация
    accuracy = scores.mean()

    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_k = k

    print(f"k={k}, Точность: {accuracy:.4f}")

print(f"\nНаилучшее значение k: {best_k}")

# Обучение модели kNN с наилучшим значением k
knn = KNeighborsClassifier(n_neighbors=best_k)
knn.fit(X_train, y_train)
# 3. Прогнозирование и оценка качества
print("\n3. Оценка качества прогноза")

# Прогнозирование на тестовой выборке
y_pred = knn.predict(X_test)

```

### 4. Оценка качества прогноза на тестовой выборке с помощью таблицы сопряженности

```

# Построение таблицы сопряженности
conf_matrix = confusion_matrix(y_test, y_pred)
confusion_df = pd.DataFrame(
    conf_matrix,

```

```
        columns=['Predicted <=50K', 'Predicted >50K'],
        index=['Actual <=50K', 'Actual >50K']
    )

print("Таблица сопряженности:")
print(confusion_df)
```

## 5. Выдать процент ошибок, допущенных классификатором на тестовой выборке

```
# 4. Вычисление процента ошибок
accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy

print(f"\n4. Процент ошибок: {error_rate * 100:.2f}%")
print(f"Точность модели: {accuracy * 100:.2f}%")

# Дополнительная информация
print(f"\nДополнительная информация:")
print(f"Правильные прогнозы для класса '<=50K': {conf_matrix[0, 0]}")
print(f"Правильные прогнозы для класса '>50K': {conf_matrix[1, 1]}")
print(f"Ложные прогнозы для класса '<=50K': {conf_matrix[0, 1]}")
print(f"Ложные прогнозы для класса '>50K': {conf_matrix[1, 0]}}")
```

**Вывод:**

## 1. Загрузка и подготовка данных...

Обучающая выборка: 32561 записей

Тестовая выборка: 16281 записей

## 2. Определение наилучшего значения k...

k=1, Точность: 0.7315

k=2, Точность: 0.7865

k=3, Точность: 0.7594

k=4, Точность: 0.7893

k=5, Точность: 0.7766

k=6, Точность: 0.7921

k=7, Точность: 0.7837

k=8, Точность: 0.7944

k=9, Точность: 0.7897

k=10, Точность: 0.7956

Наилучшее значение k: 10

## 3. Оценка качества прогноза...

Таблица сопряженности:

		Predicted <=50K	Predicted >50K
Actual <=50K	12118	317	
Actual >50K	2967	879	

4. Процент ошибок: 20.17%

Точность модели: 79.83%

Дополнительная информация:

Правильные прогнозы для класса '<=50K': 12118

Правильные прогнозы для класса '>50K': 879

Ложные прогнозы для класса '<=50K': 317

Ложные прогнозы для класса '>50K': 2967

Исходя из предоставленных результатов:

- наилучшее значение k: 10

- в таблице сопряженности видно, что модель сделала 12118 правильных прогнозов для класса <=50K и 879 правильных прогнозов для класса >50K, однако было 317 ложных прогнозов для класса <=50K и 2967 ложных прогнозов для >50K.

- процент ошибки составил 20.17%, это означает, что модель совершила ошибку в 20.17% случаев при классификации объектов

