УДК 004.89

Определение атмосферного давления с помощью метода k-ближайших соседей

Елфимов В.А., Клепиков П.Н.
Алтайский государственный университет, г. Барнаул
elfimov.0202@mail.ru, klepikov.math@gmail.com

Аннотация

Данная статья представляет методологию определения атмосферного давления с использованием метода k-ближайших соседей. В ней процесс сбора данных о погоде в 3 населённых пунктах. Затем подробно объясняется принцип работы метода k-ближайших соседей, который используется для прогнозирования атмосферного давления на основе близких значений измерений. Эффективность метода и точность результатов подтверждаются в экспериментальных исследованиях, где сравниваются предсказанные и реальные значения давления.

Ключевые слова: Машинное обучение, метод k-ближайших соседей, python

1. Введение

Метод k-ближайших соседей (k-NN) — метрический алгоритм в машинном обучении, который разработан Эвелином Фиксом и Джозефом Ходжесом в 1951 году и расширен Томасом Кавером в 1967 году. Он используется для классификации и регрессии, основываясь на сходстве объектов. В задаче классификации, объект присваивается классу, наиболее распространенному среди его k-ближайших соседей. В регрессии, значение свойства объекта определяется средним из значений соседей (см. [1]).

Преимущества метода включают его простоту, не требование дополнительного этапа обучения, применимость как в задачах классификации, так и в регрессии, а также возможность добавления новых данных в любое время. Однако у метода есть и недостатки, такие как зависимость от прошлых наблюдений, чувствительность к зашумленным данным и сложности с категориальными признаками [2].

Выбор значения k критичен для точности алгоритма: большие k могут привести к сглаживанию границ решений, в то время как малые k могут вызвать нестабильность. Также важно выбрать подходящую метрику расстояния, такую как евклидово расстояние или расстояние Минковского, в зависимости от особенностей данных и задачи. Чтобы сделать алгоритм более точным, нужно использовать меру расстояния. Евклидово расстояние — самая популярная метрика расстояния, для вычисления расстояний между точками данных. Однако нужно выбрать метрику расстояния в зависимости размеров имеющегося набора данных [3].

Расстояние Минковского — это параметрическая метрика на евклидовом пространстве, которую можно использовать для измерения расстояния между двумя точками x и y в n-мерном пространстве:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

Частные случаи:

- Расстояние городских кварталов (манхэттенское расстояние, метрика L_1, метрика городского квартала), где $p = 1$:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

- Евклидово расстояние, где $p = 2$:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Расстояние Чебышева, когда $p \to \infty$:

$$d(x, y) = \max(|x_i - y_i|)$$

Использование метода для решения различных задач:

В книге "Введение в машинное обучение" [1] описывается использование алгоритма регрессии k-ближайших соседей на основе обучающего набора wave. Они используют 3 точки тестового набора и делают несколько прогнозов с использованием разного количества соседей, а именно 1,3,9. После сравнивают прогнозы,полученные с помощью регрессии ближайщих соседей для различных значений n_neighbors. Увеличение числа соседей приводит к получению более сглаженных прогнозов, но при этом снижается правильность подгонки к обучающим данным.

Также можно рассмотреть метод использования k ближайших соседей для эффективного поиска больших неопределенных графов [4]. Основной акцент делается на вероятностных графах, возникающих из шумных измерений, моделей вывода и процессов обеспечения конфиденциальности. Предполагается независимость между рёбрами, хотя большинство результатов могут быть применимы к графам с коррелированными рёбрами. Представлен формат $G = (V, E, P, W)$, где V и E — множества узлов и рёбер, P - вероятности связанных с рёбрами, а W - веса. Граф рассматривается как выборочный граф G', где каждое ребро выбирается с вероятностью $p(e)$. Далее вводится понятие расстояния в вероятностных графах, расширяя концепцию кратчайших путей и случайных блужданий. Показывается, что мера расстояния между двумя узлами в графе, определенную на основе вероятностей наличия рёбер в этом графе, имеет ограничения, и в работе предлагаются статистические методы анализа распределения длин кратчайших путей для преодоления этих ограничений.

Еще один пример использования метода показывает вычисление предсказанных значений для диапазона возможных k с помощью модуля $NearestNeighbors$ из библиотеки $scikit - learn$ [5]. Используются три формулы для взвешенных величин, каждая для 20 значений k. Самые лучшие предсказания получаются при одном или двух ближайших соседях, а усреднение по большому диапазону не дает видимых улучшений. Скорее всего, это связано с разреженностью данных, в которых ближайшие соседи зачастую располагаются на большом расстоянии друг от друга. Зависимость от величины k тоже невелика. В любом случае нормализованная среднеквдратичная ошибка для тестового набора предсказаний находится в диапазоне 5%.

Ну и последний в данной статье пример использования метода. Набор данных Iris является одним из самых распространенных и широко используемых в области машинного обучения. Он содержит информацию о трех видах ирисов (setosa, versicolor и virginica) с измерениями четырех признаков: длины и ширины чашелистиков и лепестков. Метод

k-ближайших соседей в данном случае используется для классификации. При этом каждый ирис из тестовой выборки классифицируется на основе классов его k ближайших соседей из обучающей выборки. Расстояние между объектами измеряется в пространстве признаков, и объекту присваивается класс, который наиболее часто встречается среди его ближайших соседей. В наборе данных Iris метод k-NN позволяет определить класс каждого ириса на основе его числовых характеристик и схожести с другими ирисами. Этот метод широко применяется для начального знакомства с концепциями машинного обучения, так как набор данных Iris отличается небольшим размером и хорошо структурированными данными.

2. Описание набора данных

Атмосферное давление является важным показателем в метеорологии, оказывающим влияние на погодные условия. Для проведения исследования мы использовали три набора данных, полученных с сайта “rp5.ru”. Эти данные содержат информацию о погодных условиях в трех различных местах: Научном Городке, Боровихе и Барнауле.

В общей сложности в учет принимаются 28 погодных характеристик. Для анализа были выбраны следующие параметры: температура воздуха, атмосферное давление, относительная влажность, направление ветра, скорость ветра, общая облачность, текущая погода, дальность видимости и температура точки росы.

Таблица 1

Погодные характеристики и соответствующие им названия атрибутов в наборе данных.

Столбец	Погодная характеристика
T	Температура воздуха (градусы Цельсия) на высоте 2 метра над поверхностью земли
Po	Атмосферное давление на уровне станции (миллиметры ртутного столба)
P	Атмосферное давление, приведенное к среднему уровню моря (миллиметры ртутного столба)
Pa	Барическая тенденция: изменение атмосферного давления за последние три часа (миллиметры ртутного столба)
U	Относительная влажность (%) на высоте 2 метра над поверхностью земли
DD	Направление ветра (румбы) на высоте 10-12 метров над земной поверхностью, осредненное за 10-минутный период, непосредственно предшествовавший сроку наблюдения
Ff	Скорость ветра на высоте 10-12 метров над земной поверхностью, осредненная за 10-минутный период, непосредственно предшествовавший сроку наблюдения (метры в секунду)
ff10	Максимальное значение порыва ветра на высоте 10-12 метров над земной поверхностью за 10-минутный период, непосредственно предшествующий сроку наблюдения (метры в секунду)
ff3	Максимальное значение порыва ветра на высоте 10-12 метров над земной поверхностью за период между сроками (метры в секунду)

	Продолжение таблицы 1
N	Общая облачность
WW	Текущая погода
W1	Прошедшая погода между сроками наблюдения 1
W2	Прошедшая погода между сроками наблюдения 2
Tn	Минимальная температура воздуха за прошедший период (не более 12 часов)
Tx	Максимальная температура воздуха за прошедший период (не более 12 часов)
Cl	Слоисто-кучевые, слоистые, кучевые и кучево-дождевые облака
Nh	Количество всех наблюдающихся облаков Cl или, при отсутствии облаков Cl, количество всех наблюдающихся облаков Cm
H	Высота основания самых низких облаков (м)
Cm	Высококучевые, высокослоистые и слоисто-дождевые облака
Ch	Перистые, перисто-кучевые и перисто-слоистые облака
VV	Горизонтальная дальность видимости (км)
Td	Температура точки росы на высоте 2 метра над поверхностью земли
RRR	Количество выпавших осадков (миллиметры)
tR	Период времени, за который накоплено указанное количество осадков (часы)
E	Состояние поверхности почвы без снега или измеримого ледяного покрова
Tg	Минимальная температура поверхности за ночь (градусы Цельсия)
E’	Состояние поверхности почвы со снегом или измеримым ледяным покровом
sss	Высота снежного покрова (см)

Суммарно 3 архива имеют 73895 записей погодных условий.

3. Обработка набора данных

Первым шагом было подключение необходимых библиотек в языке программирования Python, таких как numpy, pandas, scikit-learn и matplotlib. Три набора данных были объединены в один. Затем была проведена очистка данных, удалив строки с пропущенными значениями. Для оцифровки текстовых данных, таких как направление ветра, общая облачность и текущая погода, использован LabelEncoder.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsRegressor
import re
```

```
pd.set_option('display.max_columns', None)

df1 = pd.read_csv("/content/drive/MyDrive/coursework2/Nauch_gorod.csv",
sep=r';', encoding='utf-8', on_bad_lines='skip', engine='python')

df2 = pd.read_csv("/content/drive/MyDrive/coursework2/Boroviha.csv",
sep=r',', encoding='utf-8', on_bad_lines='skip', engine='python')

df3 = pd.read_csv("/content/drive/MyDrive/coursework2/Barnaul.csv",
sep=r',', encoding='utf-8', on_bad_lines='skip', engine='python')

frames = [df1, df2, df3]
df = pd.concat(frames)

df = df[~df['P'].isnull()]
df = df[['T', 'P', 'U', 'DD', 'Ff', 'N', 'WW', 'VV', 'Td']]

Missing_Value = df.isnull().sum().sort_values(ascending=False)
Missing_Percent = (df.isnull().sum() /
df.isnull().count() * 100).sort_values(ascending=False)
Missing_Data = pd.concat([Missing_Value, Missing_Percent], axis=1,
keys=['Пропущенные значения', 'Процент'])
Missing_Data.reset_index()

df = df.dropna()

le = LabelEncoder()
le = df['DD'] = df['DD'].astype(str)
le.fit(df['DD'])
df['DD'] = le.transform(df['DD'])

df['N'] = df['N'].astype(str)
le.fit(df['N'])
df['N'] = le.transform(df['N'])

df['VV'] = df['VV'].astype(str)
le.fit(df['VV'])
df['VV'] = le.transform(df['VV'])

df['WW'] = df['WW'].astype(str)
le.fit(df['WW'])
df['WW'] = le.transform(df['WW'])
```

4. Обучение модели

Чтобы обучить набор данных, для начала разделим его значения на обучающие и тестовые в соотношении 75% и 25%:

```
x = df.drop(['P'], axis=1)
```

```
y = df['P']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

Далее обучим саму модель [6]:

```
pipe = Pipeline([('scaler', StandardScaler()), ('knr',
KNeighborsRegressor())])
params = {'knr__weights': ['uniform', 'distance'],
'knr__n_neighbors': range(1, 30 + 1), 'knr__p': range(1,5+1)}
grid = GridSearchCV(pipe, params, n_jobs=-1)
grid.fit(x_train, y_train)
print(f'Метод k ближайших соседей:')
print(f'Обучающая точность {grid.score(x_train, y_train):.3f}')
print(f'Тестовая точность {grid.score(x_test, y_test):.3f}')
print('Лучшие значения параметров:')
for param, val in grid.best_params_.items():
    param = re.match(r'\w+?__(\w+)', param)[1]
    print(f'{param} = {val}')
```

```
Метод k ближайших соседей:
Обучающая точность 0.718
Тестовая точность 0.705
Лучшие значения параметров:
n_neighbors = 28
p = 1
weights = uniform
```

Модель имеет неплохую точность, однако имеется возможность ее улучшить.

```
weights = 1/df.groupby('P')['P'].transform('count')
sample_df = df.sample(frac=1, replace=True,weights=weights)
```

Таким образом, переменная sample_df будет содержать случайную выборку из всех строк нашего набора данных df, где вероятность выбора каждой строки будет пропорциональна её весу, который обратно пропорционален количеству элементов в столбце P, к которой она принадлежит.

Обучим заново нашу модель, но с измененным набором.

```
x = sample_df.drop(['P'], axis=1)
y = sample_df['P']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
pipe = Pipeline([('scaler', StandardScaler()), ('knr',
KNeighborsRegressor())])
params = {'knr__weights': ['uniform', 'distance'],'knr__n_neighbors':
range(1, 30 + 1),'knr__p': range(1,5+1)}
grid = GridSearchCV(pipe, params, n_jobs=-1)
grid.fit(x_train, y_train)
print(f'Метод k ближайших соседей:')
print(f'Обучающая точность {grid.score(x_train, y_train):.3f}')
print(f'Тестовая точность {grid.score(x_test, y_test):.3f}')
print('Лучшие значения параметров:')
for param, val in grid.best_params_.items():
    param = re.match(r'\w+?__(\w+)', param)[1]
```

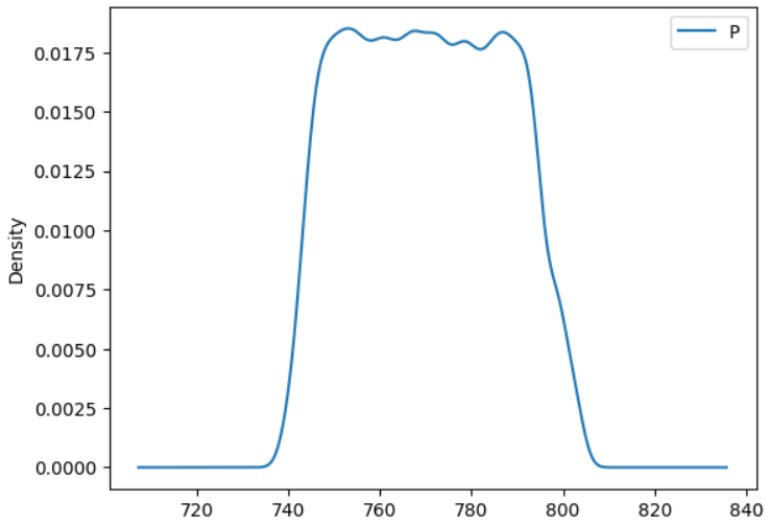

Рисунок 1. График плотности вероятности столбца P.

```
    print(f'{param} = {val}')

Метод k ближайших соседей:
Обучающая точность 0.999
Тестовая точность 0.955
Лучшие значения параметров:
n_neighbors = 22
p = 1
weights = distance
```

5. Вывод

Исследование атмосферного давления с использованием метода k-ближайших соседей предоставляет ценную информацию о возможности прогнозирования давления на основе других метеорологических данных

Список литературы

1. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python. — М. : Диалектика, 2017. — 472 с.

2. Bishop C.M. Pattern Recognition and Machine Learning. — NY : Springer, 2006.

3. Alpaydin E. Evaluation Metrics in Classification: A Comprehensive Review. — NY : Springer, 2011. — 778 p.

4. Liang Zhang, Xiang Lian, Lei Chen. Efficient k-Nearest Neighbor Search for Large Uncertain Graphs. — VLDB : Endowment, 2012.

5. Бринк Х., Ричардс Д., Феверолф М. Машинное обучение. — СПб. : Питер, 2017. — 336 с.

6. Petrou T. Pandas Cookbook. — Birmingham : Packt Publishing, 2017. — 532 p.

7. Домингос П. Верховный Алгоритм. — М. : Манн, Иванов и Фербер, 2016. — 336 с.