Министерство просвещения Российской Федерации

«Российский государственный профессионально-педагогический университет»

Кафедра информационных систем и технологий

**Курсовая работа**

по дисциплине «Математические основы машинного обучения»

Тема: «Базовые алгоритмы машинного обучения на языке Python»

Работу выполнили:

студенты гр. ПИм-301

Бирюков Д. В.,

Тихолаз Р. В.,

Хомутова А. М.

Работу проверил:

\_\_\_\_\_\_\_\_\_\_/

(подпись) (расшифровка)

Содержание

Введение 3

1. Обзор базовых алгоритмов машинного обучения 4

1.1. Линейная регрессия 4

1.1.1. Описание 4

1.1.2. Примеры применения 4

1.1.3. Реализация на Python с использованием библиотеки scikit-learn 5

1.2. Логистическая регрессия 7

1.2.1. Описание 7

1.2.2. Примеры применения 8

1.2.3. Реализация на Python с использованием библиотеки scikit-learn 8

1.3. Метод k-ближайших соседей (k-NN) 10

1.3.1. Описание 10

1.3.2. Примеры применения 11

1.3.3. Реализация на Python с использованием библиотеки scikit-learn 11

1.4. Решающие деревья 13

1.4.1. Описание 13

1.4.2. Примеры применения 14

1.4.3. Реализация на Python с использованием библиотеки scikit-learn 14

2. Оценка и сравнение результатов алгоритмов 18

2.1. Метрики оценки качества моделей 18

2.1.1. Метрики для регрессии: 18

2.1.2. Метрики для классификации: 18

2.2. Кросс-валидация для оценки обобщающей способности моделей. 19

Заключение 20

Источники 21

Введение

Машинное обучение является одной из ключевых областей в современной информатике и искусственном интеллекте. Оно представляет собой методы анализа данных, которые позволяют компьютерным системам "учиться" на основе опыта и делать прогнозы или принимать решения без явного программирования.

В сфере машинного обучения особое внимание уделяется базовым алгоритмам, которые являются фундаментальными элементами обучения. Эти алгоритмы представляют собой основу для более сложных методов и моделей и позволяют понять основные концепции и принципы машинного обучения.

Целью данной курсовой работы является изучение базовых алгоритмов машинного обучения на примере их реализации на языке программирования Python.

В этой работе мы рассмотрим несколько основных алгоритмов машинного обучения, таких как линейная и логистическая регрессия, метод k-ближайших соседей и решающие деревья. Каждый алгоритм будет описан, проиллюстрирован примерами его применения и реализован с использованием библиотеки scikit-learn.

Эта работа также предполагает анализ результатов применения алгоритмов на реальных данных и оценку их производительности с использованием различных метрик и методов оценки качества моделей.

Проведение данного исследования позволит нам получить глубокое понимание базовых алгоритмов машинного обучения и их применение на практике, что, в свою очередь, может быть полезно для разработки и реализации более сложных и эффективных моделей в дальнейшем.

1. Обзор базовых алгоритмов машинного обучения
   1. Линейная регрессия
      1. Описание

Линейная регрессия - один из самых простых и широко используемых алгоритмов машинного обучения. Она применяется для моделирования зависимостей между зависимой переменной (или целевой переменной) и одной или несколькими независимыми переменными (признаками). Модель линейной регрессии описывается уравнением:

*Y* = *β*0 + *β*1*x*1 + *β*2*x*2 + … + *βnxn* + *ϵ*

где *y* это зависимая переменная, x1, x2, …, *xn* - независимые переменные, *β*0 - свободный член, *β*1, *β*2, …, *βn* - коэффициенты регрессии, и *ϵ* - случайная ошибка.

Задача линейной регрессии заключается в нахождении таких коэффициентов *β*, которые минимизируют сумму квадратов отклонений предсказанных значений от фактических значений. Это достигается с помощью метода наименьших квадратов.

* + 1. Примеры применения

Линейная регрессия находит применение в различных областях, включая:

**Экономика**: Прогнозирование экономических показателей, таких как ВВП, уровень инфляции или безработицы на основе различных экономических факторов.

**Маркетинг**: Оценка влияния различных маркетинговых усилий на продажи продукции.

**Медицина**: Прогнозирование прогрессирования заболеваний на основе клинических данных.

**Финансы**: Моделирование и прогнозирование цен на акции и другие финансовые инструменты.

* + 1. Реализация на Python с использованием библиотеки scikit-learn

Ниже приведен пример реализации линейной регрессии на Python с использованием библиотеки scikit-learn. В этом примере мы будем использовать набор данных Iris Species, который содержит информацию о видах цветов ириса.

*LinearRegression.py:*

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import cross\_val\_predict, cross\_val\_score

from sklearn.metrics import mean\_squared\_error, r2\_score

from joblib import dump

import numpy as np

# Загрузка данных

iris = load\_iris()

X = iris.data

y = iris.target

# Создание и обучение модели линейной регрессии

model = LinearRegression()

model.fit(X, y) # Обучение модели

# Кросс-валидационное предсказание

y\_pred = cross\_val\_predict(model, X, y, cv=5)

# Оценка модели на основе кросс-валидации

cv\_scores = cross\_val\_score(model, X, y, cv=5, scoring='neg\_mean\_squared\_error')

mse\_cv = -np.mean(cv\_scores)

# Сохранение результатов предсказания

dump(model, 'linear\_regression.joblib')

# Оценка модели

mse = mean\_squared\_error(y, y\_pred)

r2 = r2\_score(y, y\_pred)

print("Среднеквадратичная ошибка: %.2f" % mse)

print("Коэффициент детерминации (R^2): %.2f" % r2)

print("Среднеквадратичная ошибка на основе кросс-валидации: %.2f" % mse\_cv)

# Визуализация результатов

plt.scatter(X[:, 0], y, color='black', label='Истинные значения')

plt.scatter(X[:, 0], y\_pred, color='blue', label='Предсказанные значения')

plt.xlabel('Длина чашелистика (см)')

plt.ylabel('Целевой показатель')

plt.title('Линейная регрессия на наборе данных Iris')

plt.legend()

plt.show()

В данном коде решается задача прогнозирования с использованием линейной регрессии на данных о ирисах. Сначала данные загружаются из набора данных Iris, затем создается и обучается модель линейной регрессии. Для оценки производительности модели используется кросс-валидация, а затем результаты сохраняются. После этого оцениваются среднеквадратичная ошибка и коэффициент детерминации. В конце, результаты визуализируются на графике, где сравниваются истинные и предсказанные значения целевой переменной в зависимости от длины чашелистика.

* 1. Логистическая регрессия
     1. Описание

Логистическая регрессия - это алгоритм машинного обучения, используемый для задач классификации, когда целевая переменная является категориальной. В отличие от линейной регрессии, которая предсказывает непрерывные значения, логистическая регрессия предсказывает вероятность принадлежности к определенному классу.

Модель логистической регрессии описывается следующим уравнением:

*P*(*y*=1∣*x*) =  
где *P*(*y*=1∣*x*) - это вероятность того, что целевая переменная *y* равна 1 при заданных значениях признаков *x.* Значения *β*0​,*β*1​,…,*βn*​ являются параметрами модели.

Для классификации бинарных данных используется логистическая функция (сигмоидальная функция), которая преобразует линейную комбинацию входных признаков в вероятность.

* + 1. Примеры применения

Логистическая регрессия находит применение в различных областях, таких как:

Медицина: Прогнозирование наличия или отсутствия заболевания на основе медицинских данных.

Финансы: Оценка кредитного риска, определение вероятности дефолта заемщика.

Маркетинг: Классификация клиентов на основе их поведения (например, вероятность покупки продукта).

Социальные сети: Определение спама в электронных сообщениях или комментариях.

* + 1. Реализация на Python с использованием библиотеки scikit-learn

*LogisticRegression.py:*

from matplotlib import pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import cross\_val\_predict

from sklearn.metrics import accuracy\_score, classification\_report

from joblib import dump

# Загрузка данных

iris = load\_iris()

X = iris.data

y = iris.target

# Создание и обучение модели логистической регрессии

model = LogisticRegression(max\_iter=1000)

model.fit(X, y) # Обучение модели

# Кросс-валидационное предсказание

y\_pred = cross\_val\_predict(model, X, y, cv=5)

# Сохранение модели

dump(model, 'logistic\_regression.joblib')

# Оценка модели

accuracy = accuracy\_score(y, y\_pred)

print("Точность: %.2f" % accuracy)

print("Отчет по классификации:")

print(classification\_report(y, y\_pred))

# Визуализация результатов

plt.figure()

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.binary, edgecolor='k', label='Истинные значения')

plt.scatter(X[:, 0], X[:, 1], c=y\_pred, cmap=plt.cm.cool, marker='x', label='Предсказанные значения')

plt.xlabel('Длина чашелистика')

plt.ylabel('Ширина чашелистика')

plt.title('Логистическая регрессия на наборе данных Iris')

plt.legend()

plt.show()

В данном коде решается задача классификации с использованием логистической регрессии на данных о ирисах. Сначала данные загружаются из набора данных Iris, затем создается и обучается модель логистической регрессии. Для оценки производительности модели используется кросс-валидация, а затем результаты сохраняются. После этого оценивается точность модели и выводится отчет по классификации. В конце, результаты визуализируются на графике, где истинные значения и предсказанные значения целевой переменной отображаются в зависимости от длины и ширины чашелистика.

* 1. Метод k-ближайших соседей (k-NN)
     1. Описание

Метод k-ближайших соседей (k-Nearest Neighbors, k-NN) - это простой и понятный алгоритм машинного обучения, используемый для задач классификации и регрессии. Основная идея метода заключается в следующем: чтобы классифицировать новый объект, алгоритм находит k ближайших объектов из обучающей выборки и присваивает новому объекту наиболее часто встречающийся класс среди этих соседей (в случае классификации) или среднее значение (в случае регрессии).

Метрика для определения ближайших соседей может быть разной, но наиболее часто используется евклидово расстояние. Количество соседей, k, является гиперпараметром модели и выбирается на основе кросс-валидации или эмпирически.

* + 1. Примеры применения

Метод k-NN применяется в различных областях, включая:

Распознавание образов: Классификация изображений и рукописных символов.

Рекомендательные системы: Рекомендация продуктов, фильмов или музыки на основе предпочтений пользователей.

Медицина: Диагностика заболеваний на основе симптомов и медицинских данных.

Финансы: Оценка кредитного риска на основе данных о клиентах.

* + 1. Реализация на Python с использованием библиотеки scikit-learn

Ниже приведен пример реализации метода k-NN на Python с использованием библиотеки scikit-learn. В этом примере мы будем использовать набор данных Iris, который содержит информацию о различных видах ирисов.

from matplotlib import pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import cross\_val\_predict

from sklearn.metrics import accuracy\_score, classification\_report

from joblib import dump

# Загрузка данных

iris = load\_iris()

X = iris.data

y = iris.target

# Создание модели k-NN

model = KNeighborsClassifier(n\_neighbors=5)

# Обучение модели на всем наборе данных

model.fit(X, y)

# Сохранение модели

dump(model, 'knn.joblib')

# Кросс-валидационное предсказание

y\_pred = cross\_val\_predict(model, X, y, cv=5)

# Оценка модели

accuracy = accuracy\_score(y, y\_pred)

print("Точность: %.2f" % accuracy)

print("Отчет по классификации:")

print(classification\_report(y, y\_pred))

# Визуализация результатов

plt.figure()

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.binary, edgecolor='k', label='Истинные значения')

plt.scatter(X[:, 0], X[:, 1], c=y\_pred, cmap=plt.cm.cool, marker='x', label='Предсказанные значения')

plt.xlabel('Длина чашелистика')

plt.ylabel('Ширина чашелистика')

plt.title('Метод k-NN на наборе данных Iris')

plt.legend()

plt.show()

Данный код выполняет задачу классификации с использованием метода k-ближайших соседей (k-NN) на наборе данных об ирисах. Сначала данные загружаются из набора данных Iris, затем создается модель k-NN с числом соседей равным 5 и обучается на всем наборе данных. Обученная модель сохраняется для дальнейшего использования. Затем выполняется кросс-валидационное предсказание для оценки производительности модели, и выводится отчет о классификации, включая точность. В конце, результаты визуализируются на графике, где истинные значения и предсказанные значения классов отображаются в зависимости от длины и ширины чашелистика.

* 1. Решающие деревья
     1. Описание

Решающие деревья - это один из популярных алгоритмов машинного обучения, используемых для задач классификации и регрессии. Они представляют собой модель, которая делает предсказания, разделяя данные на более мелкие и более однородные подмножества. Модель построена в виде дерева, где каждый узел представляет собой условие на один из признаков, ветви соответствуют результатам этого условия, а листья - конечные предсказания.

Основная идея решающих деревьев заключается в последовательном разбиении данных на подмножества, используя условия вида "признак > порог". Этот процесс продолжается до тех пор, пока не будут достигнуты определенные критерии остановки, такие как минимальное количество данных в листе или максимальная глубина дерева.

* + 1. Примеры применения

Решающие деревья широко используются в различных областях благодаря своей интерпретируемости и способности работать с данными без предварительной обработки. Вот несколько примеров применения:

Медицина: Диагностика заболеваний на основе клинических данных.

Финансы: Оценка кредитоспособности клиентов на основе их финансовых данных.

Маркетинг: Сегментация клиентов и предсказание оттока клиентов.

Экология: Прогнозирование распространения лесных пожаров или уровней загрязнения воздуха.

* + 1. Реализация на Python с использованием библиотеки scikit-learn

from matplotlib import pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.model\_selection import cross\_val\_predict

from sklearn.metrics import accuracy\_score, classification\_report

from joblib import dump

# Загрузка данных

iris = load\_iris()

X = iris.data

y = iris.target

# Создание модели решающего дерева

model = DecisionTreeClassifier()

# Обучение модели

model.fit(X, y)

# Кросс-валидационное предсказание

y\_pred = cross\_val\_predict(model, X, y, cv=5)

# Сохранение модели

dump(model, 'decisive\_trees.joblib')

# Оценка модели

accuracy = accuracy\_score(y, y\_pred)

print("Точность: %.2f" % accuracy)

print("Отчет по классификации:")

print(classification\_report(y, y\_pred))

# Визуализация результатов

plt.figure()

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.binary, edgecolor='k', label='Истинные значения')

plt.scatter(X[:, 0], X[:, 1], c=y\_pred, cmap=plt.cm.cool, marker='x', label='Предсказанные значения')

plt.xlabel('Длина чашелистика')

plt.ylabel('Ширина чашелистика')

plt.title('Решающее дерево на наборе данных Iris')

plt.legend()

plt.show()

# Визуализация дерева решений

plt.figure(figsize=(12,8))

plot\_tree(model, filled=True, feature\_names=iris.feature\_names, class\_names=iris.target\_names)

plt.title('Структура решающего дерева')

plt.show()

Представленный код реализует задачу классификации с использованием метода решающего дерева на наборе данных об ирисах. Сначала данные загружаются из набора данных Iris, затем создается модель решающего дерева и обучается на этих данных. После этого выполняется кросс-валидационное предсказание для оценки производительности модели, и выводится отчет о классификации, включая точность. Обученная модель сохраняется для дальнейшего использования. Визуализируется результат классификации на графике, где истинные значения и предсказанные значения классов отображаются в зависимости от длины и ширины чашелистика. Также представлена визуализация структуры решающего дерева.

1. Оценка и сравнение результатов алгоритмов
   1. Метрики оценки качества моделей
      1. Метрики для регрессии:

* Среднеквадратичная ошибка (MSE): MSE измеряет средний квадрат разности между предсказанными значениями и фактическими значениями. Она вычисляется по формуле:

MSE=*n*1​∑*i*=1*n*​(*yi*​−*y*^​*i*​)2

где *yi*​ - фактические значения, *y*^​*i*​ - предсказанные значения, *n* - количество наблюдений.

* Средняя абсолютная ошибка (MAE): MAE измеряет среднюю абсолютную разность между предсказанными и фактическими значениями:

MAE=*n*1​∑*i*=1*n*​∣*yi*​−*y*^​*i*​∣

* Коэффициент детерминации (R²): R² показывает, какую долю дисперсии зависимой переменной объясняет модель. Он вычисляется по формуле:

где *y*ˉ​ - среднее значение фактических значений.

* + 1. Метрики для классификации:
* Точность (Accuracy): Точность измеряет долю правильно классифицированных примеров:

где TP - истинные положительные, TN - истинные отрицательные, FP - ложные положительные, FN - ложные отрицательные.

* Precision, Recall, F1-score: Эти метрики особенно полезны при наличии несбалансированных классов:
  1. Кросс-валидация для оценки обобщающей способности моделей.

Кросс-валидация - это метод оценки качества модели, который заключается в многократном разбиении данных на обучающие и тестовые подмножества. Один из наиболее распространенных методов - k-блочная кросс-валидация (k-fold cross-validation), которая работает следующим образом:

* Данные делятся на k примерно равных подмножеств (блоков).
* Модель обучается k раз, каждый раз используя k−1 блок в качестве обучающего множества и оставшийся блок в качестве тестового множества.
* Средняя оценка качества модели рассчитывается по результатам всех k запусков.

Этот метод позволяет более точно оценить обобщающую способность модели и уменьшает вероятность переобучения.

Заключение

В данной курсовой работе мы рассмотрели базовые алгоритмы машинного обучения, такие как линейная регрессия и решающие деревья, и их реализацию на языке Python с использованием библиотеки scikit-learn. Мы начали с описания каждого алгоритма, представили примеры их применения в различных областях и подробно разобрали процесс их реализации на реальных данных.

Мы также изучили методы оценки качества моделей, такие как среднеквадратичная ошибка (MSE) для регрессии и точность (accuracy) для классификации, и обсудили важность кросс-валидации для оценки обобщающей способности моделей. Наконец, мы сравнили производительность различных алгоритмов на одном из наборов данных, что позволило выявить их сильные и слабые стороны.

Результаты наших экспериментов показали, что:

* Линейная регрессия хорошо справляется с задачами, где зависимость между переменными можно приблизительно описать линейной функцией. Этот алгоритм прост в реализации и интерпретации, но его эффективность снижается при наличии нелинейных зависимостей в данных.
* Решающие деревья являются более гибким инструментом, способным моделировать сложные нелинейные зависимости. Они легко интерпретируемы и не требуют нормализации данных. Однако решающие деревья склонны к переобучению, особенно на малых объемах данных, что требует использования методов регуляризации, таких как обрезка деревьев.

В заключение, базовые алгоритмы машинного обучения играют важную роль в анализе данных и построении моделей. Их понимание и правильное применение являются ключевыми для успешного решения задач машинного обучения и создания более сложных и эффективных моделей.

Источники:

1. Статья хабр «Машинное обучение: от Ирисов до Телекома» <https://habr.com/ru/companies/billing/articles/334738/>
2. Статься хабр «Теоретические основы всех популярных алгоритмов машинного обучения» https://habr.com/ru/articles/804605/
3. Статься medium «Cross-Validation Techniques» <https://medium.com/geekculture/cross-validation-techniques-33d389897878>
4. Проект «scikit-learn Машинное обучение в Python» https://scikit-learn.org/stable/