Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»
Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа
Дисциплина: «Машинное Обучение»
6 семестр
Задание 1

Группа:	М8О-308Б-18, №6
Студент:	Васильева Василиса Евгеньевна
Преподаватель:	Ахмед Самир Халид
Оценка:	
Дата:	

Москва, 2021

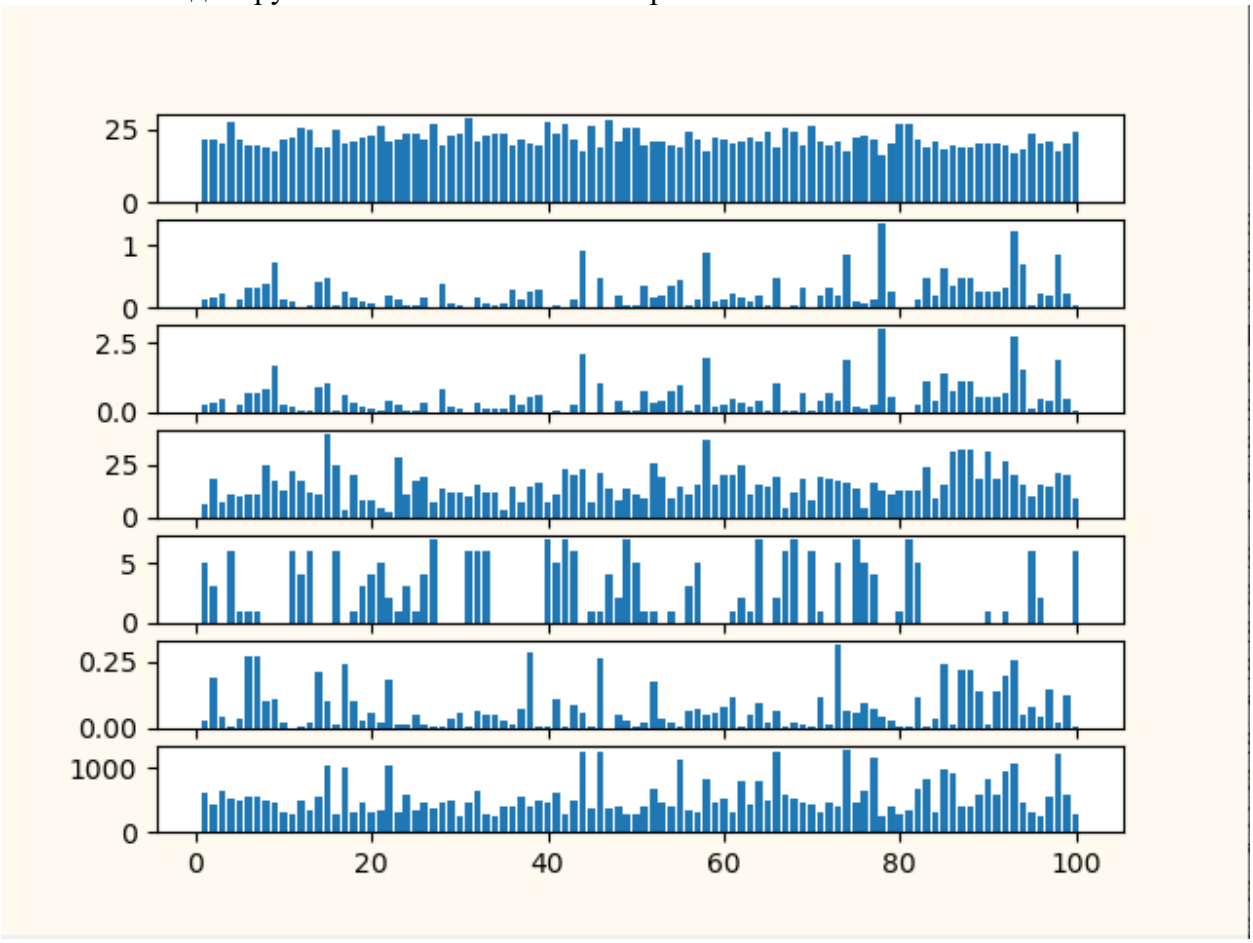
Постановка задачи

Найти себе набор данных (датасет), для следующей лабораторной работы, и проанализировать его. Выявить проблемы набора данных, устранить их. Визуализировать зависимости, показать распределения некоторых признаков. Реализовать алгоритмы К ближайших соседа с использованием весов и Наивный Байесовский классификатор и сравнить с реализацией библиотеки sklearn.

Описание датасета и препроцессинг данных

Для лабораторной работы мной был выбран датасет с ресурса kaggle, который отражает опасность метеоритов с различными параметрами (такими как скорость, размер, различные астрономические характеристики вроде Кеплеровых элементов орбиты). Всего параметров было около тридцати, поэтому необходимо было провести внимательный анализ данных и выделить самые необходимые. В ходе препроцессинга было обнаружено, что параметр диаметра повторяется три раза в разных единицах измерения, то же самое было с орбитами. Эти элементы я почистила, оставив только один параметр из нескольких, так что, можно считать, что удалила дубликаты. Некоторые параметры, которые не имели в моем понимании никакой ценности в определении опасности, я тоже убрала, таким параметром была, например, светимость объекта. В конце моей переработки данных было оставлено семь определяющих параметров.

Следующим этапом работы с данными была их визуализация. Для этого был написан небольшой код на python с использованием matplotlib.



Графики выстроены по порядку от первого к последнему параметру. Из них можем видеть, что выбросов не наблюдается.

Алгорим K ближайших соседей

Для начала работы с данными csv файл парсим и загоняем в список списков для удобного взаимодействия с данными. Суть метода в том, чтобы посмотреть k самых близко расположенных соседних элементов и выбрать класс в зависимости оттого элементы какого класса преобладают вблизи. Если алгоритм с весами, то будем складывать не единицы для выбора преобладающих классов, а веса отдельных элементов. В нашем случае используем в качестве весов единицы, деленые на евклидовы расстояния до соседей.
Реализованный KHH выдал вот такой результат на нашем датасете:

```
C:\Users\Administrator.LAPTOP-C7V2HJKO\Desktop\учебка\6 сем\machine learning\1lab>python knn.py
Accuracy: 75.14124293785311%
```

Байесовский классификатор

Байесовский классификатор использует вероятности каждого атрибута, принадлежащего каждому классу, для прогнозирования. Наивный Байес упрощает вычисление вероятностей, предполагая, что вероятность каждого атрибута, принадлежащего данному значению класса, не зависит от всех других атрибутов. Вероятность значения класса при заданном значении атрибута называется условной вероятностью. Условная вероятность считается по известной формуле Байеса. Умножая условные вероятности вместе для каждого атрибута для данного значения класса, мы получаем вероятность того, что экземпляр данных принадлежит этому классу. Чтобы сделать прогноз, можно вычислить вероятности экземпляра, принадлежащего каждому классу, и выбрать значение класса с наибольшей вероятностью.

При реализации на подготовленном датасете сначала разбиваем на классы, потом подсчитываем среднее значение атрибута для каждого класса, так же считаем стандартное отклонение для каждого атрибута класса (квадратный корень дисперсии). Имея на руках эти значения, можем считать Гауссовскую плотность вероятности. То есть, подсчитываем вероятность атрибута, принадлежащего классу. А теперь можем перемножить все вероятности всех атрибутов конкретного экземпляра, чтобы выяснить вероятность того, что весь экземпляр принадлежит классу. Чтобы предсказать класс экземпляра, выбираем наибольшую вероятность принадлежности к какому-то классу. Этот алгоритм пускаем по всему списку экземпляров, которые пытаемся предсказывать, а потом считаем, какой процент оказался верным.

Результат запуска Байеса:

```
C:\Users\Administrator.LAPTOP-C7V2HJKO\Desktop\учебка\6 сем\machine learning\1lab>python bayes.py
Accuracy: 93.26599326599326%
```

Библиотечные алгоритмы

Чтобы сравнить наши алгоритмы с библиотечными, тоже пишем новый код, где запускаем наши выборки в алгоритмы sklearn.

Их результаты:

```
C:\Users\Administrator.LAPTOP-C7V2HJKO\Desktop\учебка\6 сем\machine learning\1lab>python test.py
66.16853932584269
65.01240694789082
```

Выводы

Для меня Питон новый язык. Из-за этого возникало большое количество проблем при выполнении работы. Банально не могла импортировать файл с названием parser.py, потому что, оказалось, такой встроенный блок в Питоне есть. Алгоритмы интересные, Байес показался ощутимо сложнее, но если рассматривать его по кускам, то тоже становится понятно. Заметила, что библиотечные алгоритмы предпочитают смотреть классы в начале таблицы, а не в конце, как нам это кажется логичным. Реализация с нуля, оказалось, показывает большую точность модели, чем библиотечная.