Московский авиационный институт
(национальный исследовательский университет)
Кафедра 806
“Вычислительная математика и программирование”

Лабораторная работа №1-2
по Машинному Обучению
Вариант №2

Группа: М8О-308Б-18

Студент: Синявский А.В.

Преподаватель: Ахмед Самир Халид

Москва, 2021

Задание

Лаба 1:

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 K ближайших соседа с использованием весов и Наивный Байесовский классификатор и сравнить с реализацией библиотеки sklearn.

Лаба 2:

Необходимо реализовать алгоритмы машинного обучения. Применить данные алгоритмы на наборы данных, подготовленных в первой лабораторной работе. Провести анализ полученных моделей, вычислить метрики классификатора. Произвести тюнинг параметров в случае необходимости. Сравнить полученные результаты с моделями реализованными в scikit-learn. Аналогично построить метрики классификации. Показать, что полученные модели не переобучились. Также необходимо сделать выводы о применимости данных моделей к вашей задаче.

Вариант 2

1) Логистическая регрессия
2) Дерево решений
3) Random Forest

Алгоритмы

1. KNN

Один из простейших алгоритмов классификации. Суть алгоритма заключается в определении класса объекта по классам K уже классифицированных объектов, ближайших к новому в смысле какой-либо метрики. Выбранная мною метрика – Евклидова норма

2. Наивный Байесовский классификатор

В основе лежит формула условной вероятности Байеса. Наивным алгоритм делает предположение о независимости признаков. Данное предположение позволяет рассчитать функции правдоподобия классов

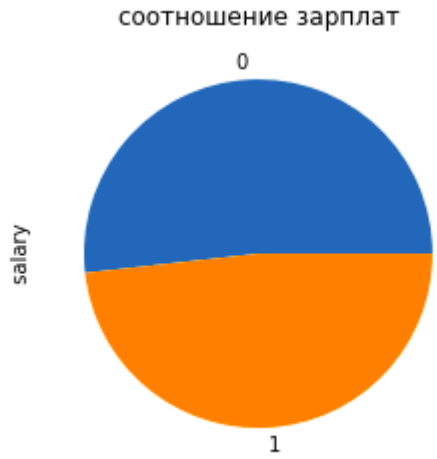
3. Логистическая регрессия

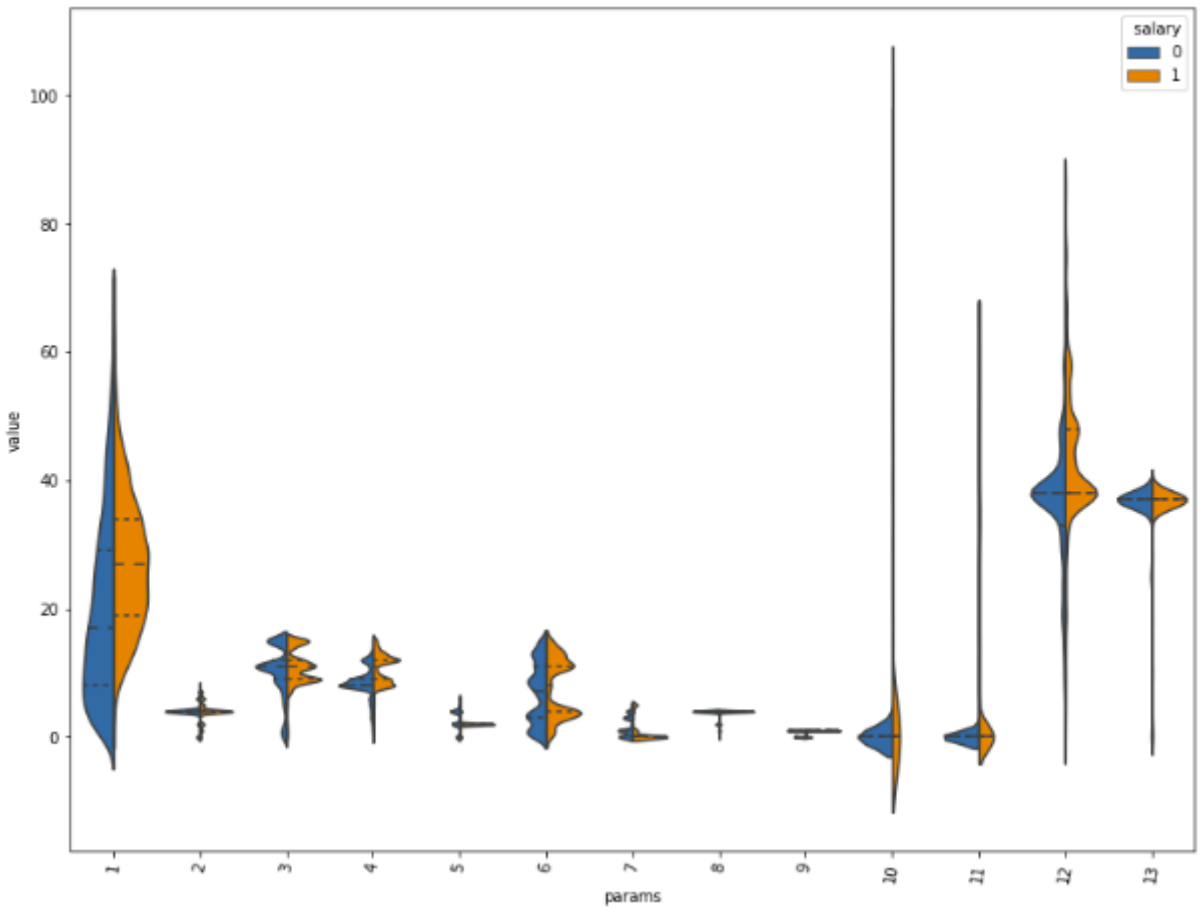
Данная модель, аналогично линейной регрессии, рассчитывает взвешенную сумму, однако определяет принадлежность объекта классу по значению сигмоидной функции:

$$y = 1 / 1 + e^{-x}$$

Данные

Используемый мною датасет – http://archive.ics.uci.edu/ml/datasets/Adult. В нём содержится информация об объёме заработной платы различых людей. Будем разделять людей на 2 группы: ЗП <=50K$, >50K$





Для оценки качества классификации рассчитывается точность предсказаний

Результаты

In [19]:
```
%%time
model = KNeighborsClassifier(n_neighbors=9)
scores = cv(model, X, y)
print("Accuracy: ", scores.mean())
```

```
Accuracy:  0.6670380687093779
CPU times: user 867 ms, sys: 128 ms, total: 995 ms
Wall time: 823 ms
```

In [20]:
```
%%time
model = bmlf.KNN(9)
scores = cv(model, X, y)
print("Accuracy: ", scores.mean())
```

```
Accuracy:  0.6670380687093779
CPU times: user 12.8 s, sys: 609 ms, total: 13.4 s
Wall time: 13.4 s
```

In [21]:
```
%%time
model = GaussianNB()
scores = cv(model, X, y)
print("Accuracy: ", scores.mean())
```

```
Accuracy:  0.7624883936861654
CPU times: user 91.3 ms, sys: 1.46 ms, total: 92.7 ms
Wall time: 92.8 ms
```

In [22]:
```
%%time
model = bmlf.NBC()
scores = cv(model, X, y)
print("Accuracy: ", scores.mean())
```

```
Accuracy:  0.7639740018570101
CPU times: user 606 ms, sys: 11.8 ms, total: 617 ms
Wall time: 611 ms
```

In [7]:
```
%%time
model = LogisticRegression()
scores = cv(model, X, y)
print("Accuracy: ", scores.mean())
```

```
Accuracy:  0.7290245669498783
CPU times: user 3.68 s, sys: 2.14 s, total: 5.82 s
Wall time: 1.59 s
```

In [18]:
```
%%time
model = bmlf.LR()
scores = cv(model, X, y)
print("Accuracy: ", scores.mean())
```

```
Accuracy:  0.5141132776230269
CPU times: user 39.1 s, sys: 27 s, total: 1min 6s
Wall time: 19.7 s
```

In [9]:
```
%%time
model = DecisionTreeClassifier()
scores = cv(model, X, y)
print("Accuracy: ", scores.mean())
```

```
Accuracy:  0.7647968759627173
CPU times: user 322 ms, sys: 4.01 ms, total: 326 ms
Wall time: 322 ms
```

In [7]:
```
%%time
model = bmlf.Node()
scores = cv(model, X, y)
print("Accuracy: ", scores.mean())
```

```
Accuracy:  0.7729101095665143
CPU times: user 7min 5s, sys: 110 ms, total: 7min 5s
Wall time: 7min 5s
```

In [15]:
```
%%time
model = RandomForestClassifier()
scores = cv(model, X, y)
print("Accuracy: ", scores.mean())
```

```
Accuracy:  0.8198546563905775
CPU times: user 5.82 s, sys: 4.11 ms, total: 5.82 s
Wall time: 5.82 s
```

In [17]:
```
%%time
model = bmlf.RF(num_trees=100)
scores = cv(model, X, y)
print("Accuracy: ", scores.mean())
```

```
Accuracy:  0.8031613203161321
CPU times: user 10.8 s, sys: 9.03 ms, total: 10.8 s
Wall time: 10.8 s
```

для Random Forest внутри алгоритма используется реализация дерева решений из sklearn, т.к. я устал ждать, пока прогонится мой неоптимизированный вариант

Выводы

По результатам работы видно, что при корректной реализации точность классификации не сильно отличается от моделей из sklearn. Основной проблемой является оптимизация. Разница во времени выполнения у большинства моделей отличается в 10 и более раз. Алгоритм KNN показал себя наименее эффективным из рассмотренных на выбранном мною датасете. Я полагаю, это связано с тем, что в алгоритме учитывается значимость различных атрибутов.