Московский авиационный институт
(Национальный исследовательский университет)
Факультет "Информационные технологии и прикладная математика"

Лабораторная работа №2
по курсу “Машинное обучение”

Студент: Живалев Е.А.

Группа: М8О-306Б

Преподаватель: Ахмед Самир Халид

Вариант: 2

Оценка: _____

Дата: _____

Москва
2021

1 Задание

Задание : 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. Задачи со звездочкой бьются по вариантам:

Вариант №2

1. Логистическая регрессия
2. Дерево решений
3. Random forest

2 Описание

2.1 Датасет

Для второй лабораторной был выбран не тот датасет, который был использован в предыдущей работе в силу того, что он имел малое количество данных и получить хорошую точность предсказывания было невозможно. Новый датасет содержит информацию о пациентах на основании которой необходимо предсказать случиться ли у пациента инсульт или нет.

При первичном изучении датасета было обнаружено, что отсутствуют данные в столбце bmi(численный признак), их было решено заменить средним значением. Также были удалены строки, где "Gender"был равен "Other".

Категориальные признаки были заменены на численные при помощи LabelEncoder из sklearn.preprocessing, численные признаки были стандартизованы с помощью StandardScaler из sklearn.preprocessing.

2.2 Логистическая регрессия

Результат

```
[30] from sklearn.metrics import accuracy_score, confusion_matrix
     lgr = LogisticRegressor()
     lgr.fit(x_train, y_train)
     predicted = lgr.predict(x_test)
     accuracy_score(predicted, y_test)

     0.9471624266144814
```

```
[31] from sklearn.linear_model import LogisticRegressionCV
     lr = LogisticRegressionCV()
     lr.fit(x_train, y_train)
     predicted = lr.predict(x_test)
     accuracy_score(predicted, y_test)

     0.9530332681017613
```

Данный результат был получен при learning rate равном 0.01 и количестве итераций равном 1000. Наиболее высокую точность, как правило, дают параметры lr = 0.01 и количество итераций равное 5000.

2.3 Дерево решений

Результат

Сравним результаты для деревьев с высотой 1, 2, 4

```
[159] depths = [1, 2, 4]
      for depth in depths:
        crt = ClassifyingDecisionTree(max_depth=depth)
        crt.fit(x_train, y_train)
        predicted = crt.predict(x_test)
        print("Depth = {}, accuracy = {}".format(depth, accuracy_score(predicted, y_test)))
```

```
Depth = 1, accuracy = 0.9530332681017613
Depth = 2, accuracy = 0.9510763209393346
Depth = 4, accuracy = 0.9510763209393346
```

```
[160] from sklearn.tree import DecisionTreeClassifier
      for depth in depths:
        sk_crt = DecisionTreeClassifier(max_depth=depth)
        sk_crt.fit(x_train, y_train)
        predicted = sk_crt.predict(x_test)
        print("Depth = {}, accuracy = {}".format(depth, accuracy_score(predicted, y_test)))
```

```
Depth = 1, accuracy = 0.9530332681017613
Depth = 2, accuracy = 0.9530332681017613
Depth = 4, accuracy = 0.9530332681017613
```

Можно видеть, что наилучшую точность дает дерево с глубиной равной 1.

2.4 Random forest

Результат

Посмотрим на результаты, когда лес состоит из 2, 5, 10, 20 деревьев

```
[186] num_trees = [2, 5, 10, 20]
      for n in num_trees:
        forest = RandomForest(num_trees=n)
        forest.fit(x_train, y_train)
        predicted = forest.predict(x_test)
        print("Number of trees = {}, accuracy = {}".format(n, accuracy_score(predicted, y_test)))
```

```
Number of trees = 2, accuracy = 0.9530332681017613
Number of trees = 5, accuracy = 0.9530332681017613
Number of trees = 10, accuracy = 0.9530332681017613
Number of trees = 20, accuracy = 0.9530332681017613
```

```
[185] from sklearn.ensemble import RandomForestClassifier

      for n in num_trees:
        forest = RandomForestClassifier(n_estimators=n)
        forest.fit(x_train, y_train)
        predicted = forest.predict(x_test)
        print("Number of trees = {}, accuracy = {}".format(n, accuracy_score(predicted, y_test)))
```

```
Number of trees = 2, accuracy = 0.9412915851272016
Number of trees = 5, accuracy = 0.9452054794520548
Number of trees = 10, accuracy = 0.9569471624266145
Number of trees = 20, accuracy = 0.9549902152641878
```

Моя реализация дает большую точность, когда лес состоит из 2 и 5 деревьев, но в отличии от реализации из библиотеки sklearn точность с увеличением количества деревьев не увеличивается.

3 Выводы

В ходе выполнения работы я познакомился с тремя алгоритмами для классификации данных. Наиболее интересным из них, на мой взгляд, является дерево решений, поскольку оно использует достаточно интуитивно понятные идеи, но при этом дает неплохой результат. Также данный алгоритм хорош тем, что явно можно видеть почему было предсказано именно то или иное значение, то есть он не является так называемым “black box” в отличие от Random Forest, который случайно бьет данные и строит, используя их, деревья решений.

4 Список литературы

1. www.kaggle.com/learn - много полезных вещей было изучено здесь
2. http://www.machinelearning.ru - описания алгоритмов
3. Ноутбуки преподавателя
4. https://scikit-learn.org/ - документация библиотеки sklearn