Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3 (3.1)

МАШИННОЕ ОБУЧЕНИЕ

Студент ИУ8-92 (Группа) Лагов С. П. (И. О. Фамилия)

Преподаватель: Коннова Н.С. (И.О. Фамилия)

2024

Цель работы:

Познакомиться с основными приемами работы с моделями классификации в scikit-learn.

Ход работы:

1. Загрузите встроенный датасет о диагностике сахарного диабета.

```
path_to_csv = '/Users/lagovs/Documents/задания на лабы/ML3.1 classification/data/diabetes.csv'
data = pd.read_csv(path_to_csv)
data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Pregnancies              768 non-null    int64  
 1   Glucose                  768 non-null    int64  
 2   BloodPressure            768 non-null    int64  
 3   SkinThickness            768 non-null    int64  
 4   Insulin                  768 non-null    int64  
 5   BMI                      768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                      768 non-null    int64  
 8   Outcome                  768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
data.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000

2. Постройте модель классификации для предсказания наличия заболевания. Оцените качество построенной модели с помощью отчета о классификации и матрицы классификации.

Доп. задание - Напишите функцию, которая автоматически обучает все перечисленные модели и для каждой выдает оценку точности.

```
accuracies = {}

def predict(model, x_train, y_train, x_test, y_test, method, show_coeffs=False):
    model.fit(x_train, y_train)
    if show_coeffs:
        print("Coefficients: \n", model.coef_)
        _ = [print(k, v) for k, v in zip(x_train.columns, model.coef_[0])]
        print("Intercept: \n", model.intercept_)

    y_pred = model.predict(x_test)

    score = metrics.accuracy_score(y_test, y_pred)
    print(f"Score: {score}")

    print(f'Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}')
    print(f'Classification Report:\n{classification_report(y_test, y_pred)}')

    global accuracies
    accuracies[method] = score
```

```
table = PrettyTable()
table.field_names = ["Method", "Precision"] # метод / оценка точности
for method in accuracies.keys():
    table.add_row([method, round(accuracies[method],8)])
print(table)
```

```
    y = data.Outcome
    X = data.drop(["Outcome"], axis=1)
    y.shape, X.shape
```

```
((768,), (768, 8))
```

```
    x_train, x_test, y_train, y_test = train_test_split(X, y, random_state=104, test_size=0.25, shuffle=True)
```

```
    predict(LogisticRegression(), x_train, y_train, x_test, y_test, "Logistic Regression", show_coeffs=False)
```

```
Score: 0.7864583333333334
Confusion Matrix:
[[112  16]
 [ 25  39]]
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.88      0.85       128
           1       0.71      0.61      0.66        64

    accuracy                           0.79       192
   macro avg       0.76      0.74      0.75       192
weighted avg       0.78      0.79      0.78       192
```

4. Постройте альтернативную полиномиальную модель, сравните ее с предыдущей.

```
poly = PolynomialFeatures(2)
x_poly_train = poly.fit_transform(x_train)
x_poly_test = poly.fit_transform(x_test)

predict(LogisticRegression(), x_poly_train, y_train, x_poly_test, y_test, "Polynomial Regression", show_coeffs=False)
```

```
Score: 0.6927083333333334
Confusion Matrix:
[[100  28]
 [ 31  33]]
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.78      0.77       128
           1       0.54      0.52      0.53        64

    accuracy                           0.69       192
   macro avg       0.65      0.65      0.65       192
weighted avg       0.69      0.69      0.69       192
```

Вывод: полиномиальная модель более точно предсказывает значения, чем линейная, это видно по значением параметра score.

5. Попробуйте применить к той же задаче другие модели регрессии. Для каждой из них выведите матрицу классификации и оценку точности.

Использовались следующие модели:

1. SVM (Метод опорных векторов) (RBF — Radial Basis Function)
2. SVM (Метод опорных векторов) (linear)
3. SVM (Метод опорных векторов) (poly)
4. Метод ближайших соседей
5. Дерево решений
6. Случайный лес
7. Многослойный перцерптрон

```
predict(svm.SVC(kernel="rbf"), x_train, y_train, x_test, y_test, "SVM (Метод опорных векторов) (RBF — Radial Basis Function)", show_coeffs=False)
```

```
Score: 0.7760416666666666
Confusion Matrix:
[[115  13]
 [ 30  34]]
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.90      0.84       128
           1       0.72      0.53      0.61        64

    accuracy                           0.78       192
   macro avg       0.76      0.71      0.73       192
weighted avg       0.77      0.78      0.77       192
```

```
predict(svm.SVC(kernel="linear"), x_train, y_train, x_test, y_test, "SVM (Метод опорных векторов) (linear)", show_coeffs=False)
```

```
Score: 0.8020833333333334
Confusion Matrix:
[[113  15]
 [ 23  41]]
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.88      0.86       128
           1       0.73      0.64      0.68        64

    accuracy                           0.80       192
   macro avg       0.78      0.76      0.77       192
weighted avg       0.80      0.80      0.80       192
```

```
predict(svm.SVC(kernel="poly"), x_train, y_train, x_test, y_test, "SVM (Метод опорных векторов) (poly)", show_coeffs=False)
```

```
Score: 0.7760416666666666
Confusion Matrix:
[[117  11]
 [ 32  32]]
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.91      0.84       128
           1       0.74      0.50      0.60        64

    accuracy                           0.78       192
   macro avg       0.76      0.71      0.72       192
weighted avg       0.77      0.78      0.76       192
```

```
predict(KNeighborsClassifier(), x_train, y_train, x_test, y_test, "KNeighborsClassifier", show_coeffs=False)
```

```
Score: 0.7239583333333334
Confusion Matrix:
[[103  25]
 [ 28  36]]
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.80      0.80       128
           1       0.59      0.56      0.58        64

    accuracy                           0.72       192
   macro avg       0.69      0.68      0.69       192
weighted avg       0.72      0.72      0.72       192
```

```
predict(DecisionTreeClassifier(), x_train, y_train, x_test, y_test, "Decision Tree", show_coeffs=False)
```

```
Score: 0.6666666666666666
Confusion Matrix:
[[95 33]
 [31 33]]
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.74      0.75       128
           1       0.50      0.52      0.51        64

    accuracy                           0.67       192
   macro avg       0.63      0.63      0.63       192
weighted avg       0.67      0.67      0.67       192
```

```
predict(MLPClassifier(), x_train, y_train, x_test, y_test, "MLP (Многослойный перцерптрон)", show_coeffs=False)
```

```
Score: 0.7239583333333334
Confusion Matrix:
[[109  19]
 [ 34  30]]
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.85      0.80       128
           1       0.61      0.47      0.53        64

    accuracy                           0.72       192
   macro avg       0.69      0.66      0.67       192
weighted avg       0.71      0.72      0.71       192
```

Таблица со значениями коэффициента детерминации:

Method	Precision
Logistic Regression	0.78645833
Polynomial Regression	0.69270833
SVM (Метод опорных векторов) (RBF — Radial Basis Function)	0.77604167
SVM (Метод опорных векторов) (linear)	0.80208333
SVM (Метод опорных векторов) (poly)	0.77604167
KNeighborsClassifier	0.72395833
Decision Tree	0.66666667
Random Forest	0.74479167
MLP (Многослойный перцерптрон)	0.72395833

Из всех моделей лучше всего себя показала модель по методу опорных векторов (linear)

Ответы на контрольные вопросы

1. Чем отличается применение разных моделей классификации в бибилиотеке _sklearn_?

   Ответ: в библиотеке представлены различные модели классификации, такие как логистическая регрессия, деревья решений и ансамблевые методы, каждая из которых имеет свои особенности и подходит для разных типов задач и данных. Выбор модели зависит от характера данных, требуемой интерпретируемости и производительности, поэтому рекомендуется тестировать несколько моделей для нахождения наилучшего решения

2. Что показывает метрика точности регрессии?

   Ответ: метрика точности регрессии показывает, насколько предсказанные значения модели близки к реальным значениям целевой переменной. Она оценивает способность модели объяснять вариацию данных и минимизировать ошибку предсказания. В зависимости от используемой метрики (MSE, MAE, R^2) точность может выражаться в виде среднего отклонения, суммы квадратов ошибок или доли объяснённой дисперсии

3. Какое значение имеют коэффициенты логистической регрессии?

   Ответ: коэффициенты логистической регрессии отражают влияние каждого признака на вероятность отнесения объекта к определённому классу. Они показывают, насколько изменение значения признака на одну единицу изменяет логарифм шансов принадлежности к классу, при условии, что остальные признаки остаются неизменными. Положительный коэффициент увеличивает шансы, отрицательный — уменьшает

4. Что показывает матрица классификации?

   Ответ: матрица классификации показывает, как модель предсказывает классы для каждого наблюдения по сравнению с их истинными значениями. Она содержит информацию о количествах верных и неверных предсказаний для каждого класса, отображая, сколько объектов было правильно классифицировано и сколько ошибочно. Это позволяет оценить качество модели, выявить смещения и анализировать, какие классы модель путает между собой

5. Какие параметры имеет конструктор объекта логистической регрессии?

   Ответ: основные — penalty, который определяет тип регуляризации (L1, L2, ElasticNet или None), и C, который настраивает силу регуляризации. Также присутствует параметр solver, определяющий метод оптимизации (например, ‘liblinear’, ‘lbfgs’, ‘newton-cg’). Параметр max_iter задаёт максимальное количество итераций для сходимости алгоритма, а random_state обеспечивает воспроизводимость результатов. Дополнительно можно настроить параметры fit_intercept, чтобы определять, учитывать ли константу в модели, и class_weight, чтобы учесть дисбаланс классов

6. Какие атрибуты имеет объект логистической регрессии?

   Ответ: coef_ содержит коэффициенты модели для каждого признака после обучения. intercept_ представляет свободный член

(смещение) модели. Атрибут classes_ содержит список классов, которые модель может предсказывать. n_iter_ показывает количество итераций, выполненных алгоритмом для достижения сходимости. Также может присутствовать score

7. Какие параметры и атрибуты имеют объекты других моделей машинного обучения библиотеки _sklearn_?

   Ответ: основные параметры включают гиперпараметры обучения, такие как регуляризация (C, penalty), метод оптимизации (solver), количество итераций (max_iter) и вес классов (class_weight). Атрибуты включают обученные коэффициенты (coef_), смещение (intercept_), список классов (classes_) и количество итераций (n_iter_)