Московский государственный технический

университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №5

Линейные модели, SVM и деревья решений

Группа: ИУ5-62Б

Студент: Селедкина А.С.

Преподаватель: Гапанюк Ю.Е.

Москва, 2020 г.

Цель лабораторной работы: изучение линейных моделей, SVM и деревьев решений.

Описание задания

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите следующие модели:
   - одну из линейных моделей;
   - SVM;
   - дерево решений.
5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.

Текст программы и примеры выполнения

Будем использовать датасет по определению наличия сердечного заболевания у пациента: https://www.kaggle.com/ronitf/heart-disease-uci.

```
data = pd.read_csv('data/heart.csv')
data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
data.shape
```

```
(303, 14)
```

```
data.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
data.dtypes
```

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

```
data.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Разделение выборки

```
x = data[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal']]
y = data['target']
```

```
# С использованием метода train_test_split разделим выборку на обучающую и тестовую
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=1)
print("x_train:", x_train.shape)
print("x_test:", x_test.shape)
print("y_train:", y_train.shape)
print("y_test:", y_test.shape)
```

```
x_train: (227, 13)
x_test: (76, 13)
y_train: (227,)
y_test: (76,)
```

Логистическая регрессия

```
log_regression = LogisticRegression(max_iter=1000)
```

```
log_regression.fit(x_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
y_predicted_lr = log_regression.predict(x_test)
```

```
precision_score(y_test, y_predicted_lr)
```

```
0.7555555555555555
```

```
recall_score(y_test, y_predicted_lr)
```

```
0.8292682926829268
```

```
classification_report(y_test, y_predicted_lr, output_dict = True)
```

```
{'0': {'precision': 0.7741935483870968,
  'recall': 0.6857142857142857,
  'f1-score': 0.7272727272727272,
  'support': 35},
 '1': {'precision': 0.7555555555555555,
  'recall': 0.8292682926829268,
  'f1-score': 0.7906976744186047,
  'support': 41},
 'accuracy': 0.7631578947368421,
 'macro avg': {'precision': 0.7648745519713261,
  'recall': 0.7574912891986063,
  'f1-score': 0.7589852008456659,
  'support': 76},
 'weighted avg': {'precision': 0.7641388417279759,
  'recall': 0.7631578947368421,
  'f1-score': 0.7614888171803716,
  'support': 76}}
```

SVM

```
SVC_model = SVC()
```

```
SVC_model.fit(x_train, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_predicted_svc = SVC_model.predict(x_test)
```

```
precision_score(y_test, y_predicted_svc)
```

```
0.5961538461538461
```

```
recall_score(y_test, y_predicted_svc)
```

```
0.7560975609756098
```

```
classification_report(y_test, y_predicted_svc, output_dict= True)
```

```
{'0': {'precision': 0.5833333333333334,
  'recall': 0.4,
  'f1-score': 0.4745762711864407,
  'support': 35},
 '1': {'precision': 0.5961538461538461,
  'recall': 0.7560975609756098,
  'f1-score': 0.6666666666666667,
  'support': 41},
 'accuracy': 0.5921052631578947,
 'macro avg': {'precision': 0.5897435897435898,
  'recall': 0.578048780487805,
  'f1-score': 0.5706214689265537,
  'support': 76},
 'weighted avg': {'precision': 0.5902496626180838,
  'recall': 0.5921052631578947,
  'f1-score': 0.5782039845376152,
  'support': 76}}
```

Дерево решений

```
decision_tree = DecisionTreeClassifier(random_state=1)
```

```
decision_tree.fit(x_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=1, splitter='best')
```

```
y_predicted_dt = decision_tree.predict(x_test)
```

```
precision_score(y_test, y_predicted_dt)
```

```
0.7941176470588235
```

```
recall_score(y_test, y_predicted_dt)
```

```
0.6585365853658537
```

```
classification_report(y_predicted_dt, y_test, output_dict= True)
```

```
{'0': {'precision': 0.8,
  'recall': 0.6666666666666666,
  'f1-score': 0.7272727272727272,
  'support': 42},
 '1': {'precision': 0.6585365853658537,
  'recall': 0.7941176470588235,
  'f1-score': 0.72,
  'support': 34},
 'accuracy': 0.7236842105263158,
 'macro avg': {'precision': 0.7292682926829268,
  'recall': 0.7303921568627451,
  'f1-score': 0.7236363636363636,
  'support': 76},
 'weighted avg': {'precision': 0.7367137355584082,
  'recall': 0.7236842105263158,
  'f1-score': 0.7240191387559808,
  'support': 76}}
```

В целом по метрикам precision и recall лучшей моделью оказалась логистическая регрессия.

```
#Важность признаков
importance = list(zip(data.columns,decision_tree.feature_importances_))
importance_sort = sorted(importance, key=itemgetter(1), reverse = True)
importance_sort
```

```
[('cp', 0.3096762702760452),
 ('exang', 0.13003799565175514),
 ('trestbps', 0.09481001256686424),
 ('ca', 0.09454693157973464),
 ('age', 0.07946029463802715),
 ('chol', 0.07765026952898858),
 ('oldpeak', 0.07643765266550838),
 ('sex', 0.056021655641918716),
 ('thalach', 0.03565104047146093),
 ('thal', 0.03237793648486447),
 ('restecg', 0.013329940494832446),
 ('fbs', 0.0),
 ('slope', 0.0)]
```

```
decision_tree2 = DecisionTreeClassifier(random_state=1)
decision_tree2.fit(x_train[['cp']],y_train)
decision_tree2.score(x_test[['cp']],y_test)
```

```
0.6973684210526315
```

```
y_predicted_dt2 = decision_tree2.predict(x_test[['cp']])
classification_report(y_predicted_dt2, y_test, output_dict= True)
```

```
{'0': {'precision': 0.7142857142857143,
  'recall': 0.6578947368421053,
  'f1-score': 0.684931506849315,
  'support': 38},
 '1': {'precision': 0.6829268292682927,
  'recall': 0.7368421052631579,
  'f1-score': 0.7088607594936709,
  'support': 38},
 'accuracy': 0.6973684210526315,
 'macro avg': {'precision': 0.6986062717770035,
  'recall': 0.6973684210526316,
  'f1-score': 0.6968961331714929,
  'support': 76},
 'weighted avg': {'precision': 0.6986062717770034,
  'recall': 0.6973684210526315,
  'f1-score': 0.696896133171493,
  'support': 76}}
```

```
decision_tree3 = DecisionTreeClassifier(random_state=1)
decision_tree3.fit(x_train[['exang']],y_train)
decision_tree3.score(x_test[['exang']],y_test)
```

```
0.6578947368421053
```

```
y_predicted_dt3 = decision_tree3.predict(x_test[['exang']])
classification_report(y_predicted_dt3, y_test, output_dict= True)
```

```
{'0': {'precision': 0.4857142857142857,
  'recall': 0.68,
  'f1-score': 0.5666666666666667,
  'support': 25},
 '1': {'precision': 0.8048780487804879,
  'recall': 0.6470588235294118,
  'f1-score': 0.717391304347826,
  'support': 51},
 'accuracy': 0.6578947368421053,
 'macro avg': {'precision': 0.6452961672473868,
  'recall': 0.6635294117647059,
  'f1-score': 0.6420289855072463,
  'support': 76},
 'weighted avg': {'precision': 0.6998899688245004,
  'recall': 0.6578947368421053,
  'f1-score': 0.667810831426392,
  'support': 76}}
```