Лабораторная работа № 18 «Презентация разработанной модели машинного обучения»

ВЫПОЛНИЛИ:

ЕФАНОВА А.Д. ШИФТ:195133

КОРОВКИНА А.С. ШИФР:195147

Постановка задачи

Цель: предсказать победителя в онлайн-игре Dota 2

Предметная область: Игра Dota 2

Dota 2 — многопользовательская компьютерная игра жанра MOBA. Игроки играют между собой матчи. В каждом матче участвует две команды, 5 человек в каждой. Одна команда играет за светлую сторону (The Radiant), другая — за тёмную (The Dire). Цель каждой команды — уничтожить главное здание базы противника (трон).

За основу было взято следующее задание: https://cutt.ly/9jSQ79C

Набор данных был сделан на основе выгрузки YASP 3.5 Million Data Dump реплеев матчей Dota 2 с сайта yasp.co.

Параметры модели (общие параметры):

Описание матча:

"match_id": 247, # идентификатор матча

"start_time": 1430514316, # дата/время начала матча, unixtime

"lobby_type": 0, # тип комнаты, в которой собираются игроки

(расшифровка в dictionaries/lobbies.csv)

Параметры модели (параметры героев):

```
# стадия выбора героев
   "picks_bans": [
      {
         "order": 0,       # порядковый номер действия
         "is_pick": false, # true если команда выбирает героя, false — если банит
         "team": 1,        # команда, совершающая действие (0 — Radiant, 1 — Dire)
         "hero_id": 95     # герой, связанный с действием
                           #    (расшифровка в dictionaries/heroes.csv)
      },
```

Выбор модели

Так как результат предсказания исхода матча является бинарным, то была как метод решения была выбрана логистическая регрессия.

Логистическая регрессия — это алгоритм классификации машинного обучения, используемый для прогнозирования вероятности категориальной зависимой переменной. В логистической регрессии зависимая переменная является бинарной переменной, содержащей данные, закодированные как 1 (да, успех и т.п.) или 0 (нет, провал и т.п.).

Линейные методы работают гораздо быстрее композиций деревьев, поэтому кажется разумным воспользоваться именно ими для ускорения анализа данных. Одним из наиболее распространенных методов для классификации является логистическая регрессия.

В качестве метрики качества будем использовать AUC-ROC (Area Under ROC-Curve). Она предназначена для алгоритмов бинарной классификации, выдающих оценку принадлежности объекта к одному из классов. По сути, значение этой метрики является агрегацией показателей качества всех алгоритмов, которые можно получить, выбирая какой-либо порог для оценки принадлежности.

Используем данные о первых пяти минутах матча игры Dota, чтобы предсказать его исход, то есть определить команду победителя. Необходимые признаки записаны в файле `features.csv`:

```
1 data = pd.read_csv('./features.csv', index_col='match_id')
2 data.head()
```

match_id	start_time	lobby_type	r1_hero	r1_level	r1_xp	r1_gold	r1_lh	r1_kills	r1_deaths	r1_items	...	dire_boots_count
0	1430198770	7	11	5	2098	1489	20	0	0	7	...	4
1	1430220345	0	42	4	1188	1033	9	0	1	12	...	4
2	1430227081	7	33	4	1319	1270	22	0	0	12	...	4
3	1430263531	1	29	4	1779	1056	14	0	0	5	...	4
4	1430282290	7	13	4	1431	1090	8	1	0	8	...	3

5 rows × 108 columns

Целевая переменная `radiant_win`. Удалим признаки, связанные с итогами матча (они помечены в описании данных как отсутствующие в тестовой выборке):

```
data.drop(['duration', 'tower_status_radiant', 'tower_status_dire', 'barracks_status_radiant', 'ba
y_train = data["radiant_win"]
X_train = data.drop('radiant_win', axis=1)
```

Приведем признаки к единому масштабу:

```
X_train.fillna(0, inplace = True)

scaler = StandardScaler()
scaler.fit(X_train)
X_normalized = scaler.transform(X_train)
X_normalized = pd.DataFrame(X_normalized, columns=X_train.columns, index = X_train.index)
```

Зафиксируем генератор разбиений для кросс-валидации по 5 блокам (`KFold`), перемешаем при этом выборку (`shuffle=True`), поскольку данные в таблице отсортированы по времени, и без перемешивания можно столкнуться с нежелательными эффектами при оценивании качества

```
1 cv = KFold(n_splits=5, shuffle=True, random_state=0)
```

Построим модель логистической регресии. Подберем налучший параметр регуляризации `C` и оценим полученное качество с помощью кросс-валидации

Наилучшее значение показателя cross_val_score достигается при C = 0.01 и равно 0.7162556488396082

```
C: 0.0001
Score: 0.7111598481541285
C: 0.001
Score: 0.7160945676853363
C: 0.01
Score: 0.7162556488396082
C: 0.1
Score: 0.7162278323009181
C: 1.0
Score: 0.7162238717590099
C: 10.0
Score: 0.716223198504357
C: 100.0
Score: 0.7162229951346128
C: 1000.0
Score: 0.7162229972634537
```

Среди признаков в выборке есть категориальные, которые мы использовали как числовые. Категориальных признаков в этой задаче одиннадцать. Попробуем убрать их из выборки и провести кросс-валидацию для логистической регрессии на новой выборке с подбором лучшего параметра регуляризации.

```
1  X_norm_drop = X_normalized.drop(['lobby_type', 'r1_hero','r2_hero','r3_hero'
```

```
Score: 0.7161305987610914
C: 0.01
Score: 0.7163172841723874
C: 0.1
Score: 0.7162900089591753
C: 1.0
Score: 0.7162856526181904
C: 10.0
Score: 0.7162851146715985
C: 100.0
Score: 0.716285085017043
C: 1000.0
Score: 0.7162850405306902
```

Качество не изменилось, следовательно, удаленные признаки никак не влияли на целевую переменную.

Построим ROC-кривую, предварительно обучив модель с параметром C = 0.01, и разбив выборку на обучающую и тестовую.

Качество предсказания ухудшилось.

```
1  from sklearn.metrics import roc_auc_score
2  print ("AUC-ROC = ", roc_auc_score(y_test, y_pred))
```

```
AUC-ROC =  0.6528077175055464
```

```
1  fpr, tpr, thresholds = roc_curve(y_test, y_pred)
2  plt.plot([0,1], [0,1], linestyle='--')
3  plt.plot(fpr, tpr, marker='.')
4  plt.show()
```