Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и механики

Высшая школа Теоретической механики

Курсовая работа

Соревнование «Catch Me If You Can (Alice)»

по дисциплине «Языки программирования»

Выполнили

студенты гр.3630103/80001	*<подпись>*	О.Н. Мальцева
3630103/80002	*<подпись>*	М.М. Фролов
	<подпись>	П.А. Мухаметова
	<подпись>	Н.А. Зырянов

Руководитель *<подпись>* П.М. Кикин

«___» _____ 2020 г.

Санкт-Петербург

2020

Содержание

Введение. 3

1.Постановка задачи. 4

2.Разбор программного кода. 5

Заключение. 8

Введение

Kaggle — система организации конкурсов по исследованию данных, а также социальная сеть специалистов по обработке данных и машинному обучению. Среда организована как публичная веб-платформа, на которой пользователи и организации могут публиковать наборы данных, исследовать и создавать модели, взаимодействовать с другими специалистами по данным и инженерами по машинному обучению, организовывать конкурсы по исследованию данных и участвовать в них. В системе размещены наборы открытых данных, предоставляются облачные инструменты для обработки данных и машинного обучения. Также реализованы обучающие ресурсы, имеется раздел для размещения вакансий работодателями, где тоже возможна организация конкурсов для отбора наилучших кандидатов.

В этой среде мы будем работать над проектом «Catch Me If You Can (Alice)», который был разработан компанией Яндекс.

1.Постановка задачи

Будем решать задачу идентификации взломщика по его поведению в сети Интернет. Это сложная и интересная задача на стыке анализа данных и поведенческой психологии. В качестве примера, компания Яндекс решает задачу идентификации взломщика почтового ящика по его поведению. В двух словах, взломщик будет себя вести не так, как владелец ящика: он может не удалять сообщения сразу по прочтении, как это делал хозяин, он будет по-другому ставить флажки сообщениям и даже по-своему двигать мышкой. Тогда такого злоумышленника можно идентифицировать и "выкинуть" из почтового ящика, предложив хозяину войти по SMS-коду. Этот пилотный проект описан в статье на Хабрахабре. Похожие вещи делаются, например, в Google Analytics и описываются в научных статьях, найти можно многое по фразам "Traversal Pattern Mining" и "Sequential Pattern Mining".

В этом соревновании будем решать похожую задачу: алгоритм будет анализировать последовательность из нескольких веб-сайтов, посещенных подряд одним и тем же человеком, и определять, Элис это или взломщик (кто-то другой).

Данные собраны с прокси-серверов Университета Блеза Паскаля. "A Tool for Classification of Sequential Data", авторы Giacomo Kahn, Yannick Loiseau и Olivier Raynaud.

2.Разбор программного кода

Подключение библиотек:

```
import eli5
from collections import Counter
import gc
import re
from pathlib import Path
from scipy.sparse import csr_matrix, hstack, vstack
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', 100)

from IPython.display import display_html, display
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(palette='muted')

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler, OneHotEncoder
from sklearn.model_selection import TimeSeriesSplit, GridSearchCV, StratifiedKFold, KFold, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

import warnings
warnings.simplefilter('ignore', category=UserWarning)
warnings.simplefilter('ignore', category=FutureWarning)
```

Создание двух списков с названиями признаков:

```
SEED = 7
sites = [f'site{i}' for i in range(1, 11)]
times = [f'time{i}' for i in range(1, 11)]
```

Загрузка обучающей и тестовой выборки:

```
train_df = pd.read_csv(PATH/'train_sessions.csv', parse_dates=times, nrows=None)
print(train_df.shape)
train_df.head()
```

```
test_df = pd.read_csv(PATH/'test_sessions.csv', parse_dates=times, nrows=None)
print(test_df.shape)
test_df.head()
```

Логистическая регрессия:

```
logit = LogisticRegression(C=1., solver='liblinear', random_state=SEED)
time_cv_split = TimeSeriesSplit(n_splits=10)
skf_cv_split = StratifiedKFold(n_splits=10)
```

Функция cross_val_scheme, в которой считается crossvalscore:

```
def cross_val_scheme(model, X_train, y_train, cv_split, site_features, extra_features=None):
    cv_scores = cross_val_score(model, X_train, y_train, scoring='roc_auc', cv=cv_split, n_jobs=-1)
    print(f'CV Scores \n {cv_scores}')
    print(f'Mean cv score: {cv_scores.mean()} \t Std cv score: {cv_scores.std()}')
    model.fit(X_train, y_train)

    if extra_features is not None:
        all_features = site_features + extra_features
    else:
        all_features = site_features

    display_html(eli5.show_weights(model, feature_names=all_features, top=30))
    if extra_features is not None:
        print(pd.DataFrame({'Feature': extra_features, 'coef': model.coef_.flatten()[-len(extra_features):]}))

    return model, cv_scores
```

Функция vectorize_sites, применяющая векторизацию данных:

```
def vectorize_sites(train_df, test_df, sites_df, params):
    train_sessions = train_df[sites].fillna(0).astype(np.int32).apply(lambda row: ' '.join([sites_df[site] for site in row]), axis=1)
    test_sessions = test_df[sites].fillna(0).astype(np.int32).apply(lambda row: ' '.join([sites_df[site] for site in row]), axis=1)
    vectorizer = CountVectorizer(**params)
    train_sessions = vectorizer.fit_transform(train_sessions)
    test_sessions = vectorizer.transform(test_sessions)

    return vectorizer, train_sessions, test_sessions
```

Функция add_time_features, выделяющая признаки, связанные со временем посещения сайтов:

```
def add_time_features(df):
    tmp_df = pd.DataFrame()
    cat_feats = []

    tmp_df['start_month'] = df['time1'].dt.month
    cat_feats.append('start_month')

#     tmp_df['start_year'] = df['time1'].dt.year

    tmp_df['year_month'] = 12*(df['time1'].dt.year - 2013) + df['time1'].dt.month
    cat_feats.append('year_month')

    tmp_df['start_weekday'] = df['time1'].dt.dayofweek
    cat_feats.append('start_weekday')

    tmp_df['start_weekend'] = (tmp_df['start_weekday'] > 4).astype(np.int32)

    tmp_df['start_hour'] = df['time1'].dt.hour
    cat_feats.append('start_hour')

    # Alice is mostly active
    tmp_df['active1213'] = df['time1'].apply(lambda t: 12 <= t.hour <= 13).astype(np.int32)
    tmp_df['active1618'] = df['time1'].apply(lambda t: 16 <= t.hour <= 18).astype(np.int32)
#     tmp_df['hour915'] = df['time1'].apply(lambda x: (x.hour == 9 or x.hour == 15)).astype('int')

    # Alice not active
    tmp_df['active0911'] = df['time1'].apply(lambda t: 9 <= t.hour <= 11).astype(np.int32)

#     tmp_df['start_minute'] = df['time1'].dt.minute
#     tmp_df['minutespent'] = (df['time1'].dt.hour * 60) + df['time1'].dt.minute

#     tmp_df['morning'] = ((df['time1'].dt.hour >= 0) & (df['time1'].dt.hour <= 5)).astype(np.int32)

#     tmp_df['day'] = ((df['time1'].dt.hour >= 6) & (df['time1'].dt.hour <= 11)).astype(np.int32)

    tmp_df['evening'] = ((df['time1'].dt.hour >= 12) & (df['time1'].dt.hour <= 17)).astype(np.int32)

    tmp_df['night'] = ((df['time1'].dt.hour >= 17) & (df['time1'].dt.hour <= 23)).astype(np.int32)

#     tmp_df['is_quarter_end'] = df['time1'].dt.is_quarter_end.astype(np.int32)

    return tmp_df, cat_feats
```

One-hot encoding:

```
train_split = train_time_data.shape[0]
full_time_data = pd.get_dummies(pd.concat((train_time_data, test_time_data)), columns=cat_feats, drop_first=True)
train_oh_data = full_time_data[:train_split]
test_oh_data = full_time_data[train_split:]
```

Функция create_submission, подготавливающая данные для размещения:

```
def create_submission(test_preds, session_ids, name='submission'):
    df = pd.DataFrame({'session_id': session_ids, 'target': test_preds})
    df.to_csv(f'{name}.csv', header=True, index=False)
    return df
```

Подготавливаем данные для размещения с помощью функции create_submission:

```
logit.fit(train_data4, y_train)
test_preds = logit.predict_proba(test_data4)[:, 1]
test_sub = create_submission(test_preds, test_sessions)
```

Заключение

Таким образом, в результате проделанной работы был придуман и реализован алгоритм решения задачи идентификации взломщика по его поведению в сети Интернет “Catch Me If You Can ("Alice")” на языке Python.