گزارش پروژه پایانی داده‌کاوی

مهدیس صفری ۹۵۳۱۰۵۱

ابتدا ترتیب داده‌ها را با استفاده از تابع shuffle بهم می‌ریزیم و با استفاده از تابع feature creation ویژگی‌های جدید به dataframe اضافه می‌کنیم.

```
def feature_creation(dataframe):
    dataframe['reservation_status_year'] = pandas.DatetimeIndex(dataframe['reservation_status_date']).year
    dataframe['reservation_status_month'] = pandas.DatetimeIndex(dataframe['reservation_status_date']).month
    dataframe['reservation_status_day'] = pandas.DatetimeIndex(dataframe['reservation_status_date']).day
    dataframe.drop('reservation_status_date', inplace=True, axis=1)
    dataframe['people'] = dataframe['adults'] + dataframe['children'] + dataframe['babies']
    dataframe['not_canceled_minus_canceled'] = dataframe['previous_bookings_not_canceled'] - dataframe['previous_cancellations']
    return dataframe
```

با استفاده از ویژگی reservation status date سه ویژگی سال، ماه و روز مربوطه را به dataframe اضافه می‌کنیم. همچنین ویژگی people که تعداد نفرات را نشان می‌دهد با جمع نفرات بزرگسال، کودک و نوزاد به dataframe اضافه می‌کنیم. ویژگی دیگری که به dataframe اضافه شده است تفاضل دفعاتی است که شخص قبلا رزرو انجام داده و کنسل نکرده با تعداد کنسل شده‌ها است.

در مرحله بعد، در تابع preprocess مشابه تمرین‌های قبل تمامی مقادیر data frame را به مقادیر عددی int تبدیل می‌کنیم و درصورت وجود داده تکراری یا ستون بدون تغییر آن را حذف می‌کنیم.

سپس داده‌های موجود در data frame به دو دسته X و Y یا همان label و feature تقسیم شده و توسط توابع زیر نرمال می‌شوند.

```
sc = StandardScaler()
X = sc.fit_transform(X)
Y = Y.astype('int')
```

در ادامه به انتخاب k ویژگی مهمتر می‌پردازیم.

```
    # feature extraction
    test = SelectKBest(score_func=f_classif, k=10)
    fit = test.fit(X, Y)
    # summarize scores
    set_printoptions(precision=10)
    # print(fit.scores_)
    X_new = fit.transform(X)
```

در این قسمت از تابع feature selection، k ویژگی با بیشترین اطلاعات برای دسته‌بندی انتخاب کرده که k با توجه به امتیازهای مشاهده شده از طریق fit.scores_ تعیین شده است.

ویژگی‌های انتخاب شده بدین ترتیب هستند:

'lead_time', 'distribution_channel', 'deposit_type', 'customer_type', 'adr', 'required_car_parking_spaces', 'total_of_special_requests', 'reservation_status', 'reservation_status_month', 'people'

سپس داده‌ها با استفاده از تابع زیر که دو دسته داده‌های آموزشی و تست تقسیم می‌شوند.

```
X_train, X_test, Y_train, Y_test = train_test_split(features, label, test_size=0.3)  # 70% training and 30% test
```

در ادامه به بررسی توابع مربوط به Neural Network و Random Forest می‌پردازیم.

```
def NN():
    print("~~~~~~~~~~~~~~ Neural Network ~~~~~~~~~~~~~~~~")
    mlp = MLPClassifier(hidden_layer_sizes=(4), max_iter=100)
    mlp.fit(X_train, Y_train)
    predictions = mlp.predict(X_test)
    train_eval(predictions)
```

دسته‌بندی به روش Neural Network با استفاده از تابع بالا انجام می‌شود.

در خط اول تابع MLPClassifier دو پارامتر را بعنوان ورودی دریافت میکند. پارامتر اول تعداد لایه‌های پنهان را تعیین می‌کند. در اینجا ۱ لایه با ۴ node تعیین کردیم. که این مقدار با آزمون و خطا بدست آمد. پارامتر دوم تعداد iteration را تعیین می‌کند.

خط دوم تابع fit است که برای یاگیری الگوریتم روی داده آموزشی بکار می‌رود.

و در نهایت با استفاده از مدل بدست آمده داده‌های تست پیش‌بینی می‌شوند و دقت دسته‌بندی در تابع train eval سنجیده می‌شود که در ادامه به آن خواهیم پرداخت.

```
def RF():
    print("~~~~~~~~~~~~~~ Random Forest ~~~~~~~~~~~~~~~~")
    clf = RandomForestClassifier(n_estimators=100)
    clf.fit(X_train, Y_train)
    predictions = clf.predict(X_test)
    train_eval(predictions)
```

دسته بندی به روش Random Forest با استفاده از تابع بالا انجام می‌شود. ورودی تابع RandomForestClassifier تعداد درخت‌ها را مشخص می‌کند. در ادامه مشابه شبکه عصبی توابع fit، predict روی مدل صدا شده و نتایج بدست آمده توسط تابع train eval سنجیده می‌شود.

```python
def train_eval(predictions):
    print(classification_report(Y_test, predictions))
    print(confusion_matrix(Y_test, predictions))
    print("R2: \t", r2_score(Y_test, predictions))
    print("RMSE: \t", sqrt(mean_squared_error(Y_test, predictions)))
    print("MAE: \t", mean_absolute_error(Y_test, predictions))
```

برای ارزیابی الگوریتم، معمولا از confusion matrix، precision، recall و f1 score استفاده می‌شود که سه مورد آخر همچنین accuracy با استفاده از تابع classification report بدست می‌آید و مورد اول که ماتریسی n*n است (n تعداد کلاس-هاست.) با استفاده از تابع confusion matrix بدست می‌آید. با استفاده از این ماتریس تعداد پیش‌بینی‌های درست و نتیجه پیش-بینی‌های اشتباه را می‌توان مشاهده کرد.

نتایج :

~~~~~~~~~~~~~~~ Neural Network ~~~~~~~~~~~~~~~~~~

	precision	recall	f1-score	support
0	1.00	1.00	1.00	18938
1	1.00	1.00	1.00	7162
accuracy			1.00	26100
macro avg	1.00	1.00	1.00	26100
weighted avg	1.00	1.00	1.00	26100

```
confusion matrix :
[[18938     0]
 [    0  7162]]
R2:      1.0
RMSE:    0.0
MAE:     0.0
```

~~~~~~~~~~~~~~~ Random Forest ~~~~~~~~~~~~~~~~~~

	precision	recall	f1-score	support
0	1.00	1.00	1.00	18938
1	1.00	1.00	1.00	7162
accuracy			1.00	26100
macro avg	1.00	1.00	1.00	26100
weighted avg	1.00	1.00	1.00	26100

```
confusion matrix :
[[18938     0]
 [    0  7162]]
R2:      1.0
RMSE:    0.0
MAE:     0.0
```

Random Forest از نظر محاسباتی کم هزینه‌تر و سریع‌تر است، و برای یادگیری نیاز به GPU ندارد. همچنین برای استفاده از Neural Network لازم است برای ساخت بهترین مدل با آزمون و خطا دنبال بهترین پارامترها برای مدل باشیم و بهمین دلیل قابلیت تفسیرپذیری کمی دارد.