دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

درس هوش مصنوعی و کارگاه

گزارش ۵: تشخیص تقلب روی کارت‌های اعتباری بانکی با کمک یادگیری ماشین

نگارش

مهسا گودرزی

۹۹۱۲۰۴۳

استاد اول

دکتر مهدی قطعی

استاد دوم

بهنام یوسفی مهر

دی ۱۴۰۲

چکیده

در این مقاله، ما با اســتفاده از روش‌های داده‌کاوی خوشــه‌بندی K-means و طبقه‌بندی Logistic Regression، تقلب‌های مالی روی کارت‌های بانکی اروپا را تشـــخیص می‌دهیم و برای این کار از یک مجموعه داده که شـامل اطلاعات مربوط به تراکنش‌های کارت‌های بانکی در سـال ۲۰۱۹ اسـت، اسـتفاده می‌کنیم. الگوریتم‌های خوشـــه‌بندی و طبقه‌بندی را بر روی این مجموعه داده پیاده‌ســـازی و ارزیابی می‌کنیم و نتایج حاصل را با یکدیگر مقایسه می‌کنیم.

واژه‌های کلیدی:

خوشه‌بندی، K-means، طبقه‌بندی، Logistic Regression

فهرست مطالب

صفحه

فهرست اشکال

صفحه

فصل اول

مقدمه

مقدمه

در دنیای امروز، تقلب مالی یکی از مشکلات جدی و پیچیده‌ای است که برای کشف و جلوگیری از آن نیاز به روش‌های هوشمند و کارآمد داریم. تقلب مالی می‌تواند به شکل‌های مختلفی انجام شود، از جمله تقلب در صورت‌های مالی، تقلب در کارت‌های بانکی، تقلب در بیمه و غیره. بنابراین تشخیص تقلب مالی یکی از چالش‌های مهم و مورد توجه پژوهشگران و متخصصان حوزه‌های مختلف است.

یکی از روش‌های موثر برای تشخیص تقلب مالی استفاده از داده‌کاوی[۱] است. داده‌کاوی عبارت است از استخراج دانش و الگوهای مفید از حجم زیادی از داده‌ها با استفاده از تکنیک‌های آماری، ریاضی و محاسباتی. داده‌کاوی می‌تواند به ما کمک کند تا روابط، رفتارها و عوامل موثر بر تقلب مالی را شناسایی و پیش‌بینی کنیم. از جمله تکنیک‌های داده‌کاوی که برای تشخیص تقلب مالی کاربرد دارند، خوشه‌بندی[۲] و طبقه‌بندی[۳] هستند.

خوشه‌بندی یک روش یادگیری بدون نظارت[۴] است که به ما امکان می‌دهد تا داده‌ها را بر اساس شباهت و تفاوت‌های آن‌ها به گروه‌های مختلف تقسیم کنیم. خوشه‌بندی می‌تواند به ما کمک کند تا داده‌هایی که از الگوی عادی خارج هستند را تشخیص دهیم و به عنوان داده‌های ناهنجار یا تقلبی معرفی کنیم. یکی از روش‌های مرسوم برای خوشه‌بندی داده‌ها، روش خوشه‌بندی k-میانگین[۵] است. این روش با تعیین تعداد خوشه‌ها به صورت از پیش تعیین شده، سعی می‌کند تا داده‌ها را به گونه‌ای خوشه‌بندی کند که فاصله داده‌ها از مرکز خوشه‌ای که به آن تعلق دارند، کمینه شود.

طبقه‌بندی یک روش یادگیری تحت نظارت[۶] است که به ما امکان می‌دهد تا داده‌ها را بر اساس برچسب‌هایی که به آن‌ها اختصاص داده شده‌اند، به دسته‌های مختلف تقسیم کنیم. طبقه‌بندی می‌تواند

---

۱ Data Mining

۲ Clustering

۳ Classification

۴ Unsupervised Learning

۵ K-means

۶ Supervised Learning

به ما کمک کند تا داده‌های جدید را بر اساس ویژگی‌های آن‌ها به دسته‌های موجود دسته‌بندی کنیم و احتمال تقلبی بودن یا نبودن آن‌ها را محاسبه کنیم. یکی از روش‌های مرسوم برای طبقه‌بندی داده‌ها، روش Logistic Regression است. این روش با استفاده از یک تابع غیرخطی به نام تابع Logistic، سعی می‌کند تا احتمال تعلق داده‌ها به یکی از دو دسته موجود را پیش‌بینی کند.

در این مقاله، ما قصد داریم تا با استفاده از روش‌های خوشه‌بندی k-میانگین و Logistic Regression، تقلب‌های مالی روی کارت‌های اعتباری بانکی اروپا را تشخیص دهیم. برای این منظور، از یک مجموعه داده[۷] که شامل اطلاعات مربوط به تراکنش‌های کارت‌های بانکی در سال ۲۰۱۳ است، استفاده می‌کنیم. این مجموعه داده شامل ۲۸ ویژگی مختلف است که با استفاده از تکنیک‌های کاهش بعد، از داده‌های اصلی استخراج شده‌اند. همچنین شامل دو ویژگی به نام‌های Time و Amount است که نشان‌دهنده زمان و مبلغ تراکنش هستند. برچسب هر تراکنش نیز با مقدار ۰ یا ۱ نشان داده شده است که ۰ به معنی تراکنش عادی و ۱ به معنی تراکنش تقلبی است. ما با استفاده از زبان برنامه‌نویسی پایتون و کتابخانه‌های مربوطه، الگوریتم‌های خوشه‌بندی و طبقه‌بندی را بر روی این مجموعه داده پیاده‌سازی و ارزیابی می‌کنیم و نتایج حاصل را با یکدیگر مقایسه می‌کنیم.

[۷] Dataset

فصل دوم

خوشه‌بندی K-means

خوشه‌بندی K-means

در این فصل برای تشخیص تقلب روی کارت‌های اعتباری از روش خوشه‌بندی K-means استفاده می‌کنیم. برای این کار ابتدا Dataset را بارگیری و سپس ساختار داده و اطلاعات موجود را بررسی می‌کنیم. سپس از الگوریتم K-means استفاده می‌کنیم و به بررسی خوشه‌ها و تحلیل نتایج به دست آمده از این روش می‌پردازیم.

۲-۱- آماده‌سازی داده

برای بارگیری اطلاعات موجود در Dataset می‌توان از کتابخانه Pandas استفاده کرد. با کمک توابع مختلف این کتابخانه، ما می‌توانیم ساختار کلی داده را ببینیم؛ این مجموعه داده شامل ۲۸ ویژگی مختلف است که با استفاده از تکنیک‌های کاهش بعد، از داده‌های اصلی استخراج شده‌اند. همچنین شامل دو ویژگی به نام‌های Time و Amount است که نشان‌دهنده زمان و مبلغ تراکنش هستند. برچسب هر تراکنش نیز با مقدار ۰ یا ۱ نشان داده شده است که ۰ به معنی تراکنش عادی و ۱ به معنی تراکنش تقلبی است و این برچسب‌ها در ستون Class ذخیره شده‌اند که ۲۸۴۳۱۵ تا از برچسب‌ها مقدار ۱ و ۴۹۲ تا از برچسب‌ها مقدار ۰ را دارند. پس در کل این مجموعه داده شامل ۳۱ ستون و ۲۸۴۸۰۷ سطر است.

حال ما می‌خواهیم که نمونه‌های خود و برچسب‌ها که هدف ما برای تحلیل نتایج هستند به صورت متغیرهای جدا ذخیره کنیم. نمونه‌ها را که شامل ویژگی‌های داده هستند در متغیر basic_samples و برچسب‌ها در متغیر target_labels ذخیره می‌شوند. تصویر کد این بخش در شکل ۲-۱ آورده شده است.

```
basic_samples = data.drop(columns='Class')

target_labels = data['Class'].ravel()
```

شکل ۲-۱- تعریف متغیرهای basic_samples و target_labels

بعد از تعریف این متغیرها و مشخص کردن samples و target به نرمال‌سازی نمونه‌ها و سپس کاهش بعد آن‌ها با کمک PCA می‌پردازیم. همان‌طور که در شکل ۲-۲ آورده شده است، ما با کمک کلاس StandardScaler از کتابخانه sklearn.preprocessing نمونه‌ها را استاندارد کرده‌ایم، یعنی آنها را به گونه‌ای تغییر می‌دهیم که دارای میانگین ۰ و واریانس ۱ شوند. سپس با کمک کلاس PCA از کتابخانه sklearn.decomposition بعد داده‌ها را به ۲ کاهش می‌دهیم. در واقع PCA یک روش کاهش بعد است که برای کاهش تعداد متغیرهای یک داده بزرگ استفاده می‌شود. هدف از این کار این است که با حفظ بیشترین اطلاعات ممکن در داده، آن را به یک داده کوچک‌تر تبدیل کنیم. برای این کار، PCA از یک تبدیل خطی استفاده می‌کند که داده را به یک سیستم مختصات جدید (مؤلفه‌های اصلی) منتقل می‌کند که در آن جهت‌هایی که بیشترین تغییرات را در داده نشان می‌دهند، به راحتی قابل شناسایی هستند.

```
normalized_samples = StandardScaler().fit_transform(basic_samples)

samples_pca = PCA(n_components=2).fit_transform(normalized_samples)
```

شکل ۲-۲- نرمال‌سازی نمونه‌ها و اعمال PCA بر روی آن‌ها

با انجام تمامی این کارها، داده‌های ما برای ورود به الگوریتم K-means آماده شده‌اند.

۲-۲- الگوریتم K-means

برای پیاده‌سازی الگوریتم K-means بر روی داده‌های خود، از کتابخانه sklearn.cluster کمک می‌گیریم. کلاس KMeans با گرفتن ورودی‌های لازم، به راحتی می‌تواند خوشه‌بندی را انجام دهد. از آنجا که ما می‌خواهیم نمونه‌های خود را به دو خوشه تقسیم کنیم، ورودی n_clusters در KMeans را برابر با ۲ قرار می‌دهیم. همچنین برچسب‌های مربوط به نتایج خوشه‌بندی K-means را در متغیر kmeans_labels ذخیره می‌کنیم. تصویر کد این بخش در شکل ۲-۳ آورده شده است.

```
kmeans = KMeans(init='k-means++', n_clusters=2, n_init=10, random_state=42)
kmeans.fit(samples_pca)
kmeans_labels = kmeans.predict(samples_pca)
```

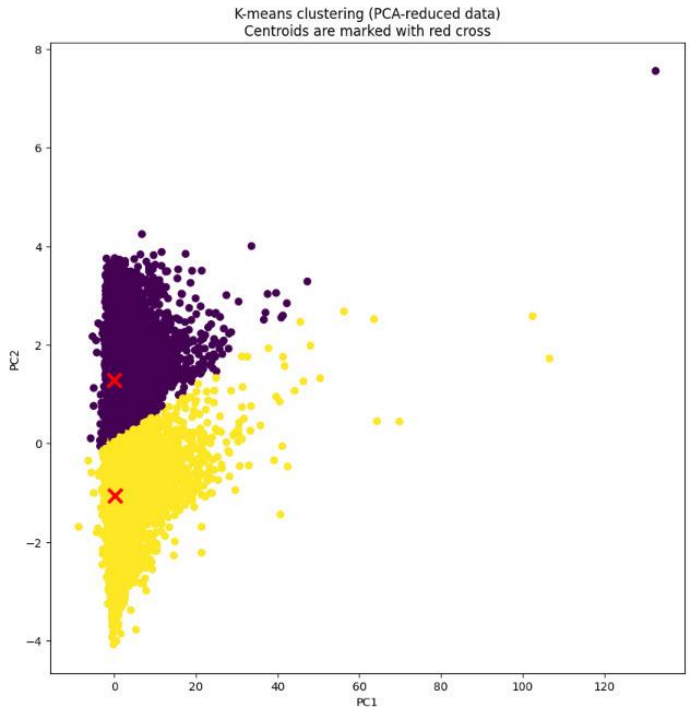
شکل ۲-۳- تصویر کد الگوریتم K-means

برای بصری‌سازی نتایج خوشه‌بندی K-means می‌توانیم از کتابخانه matplotlib.pyplot استفاده کنیم. این نمودار نحوه توزیع نمونه‌ها بر اساس PCA را نشان می‌دهد و برای مشخص کردن نحوه خوشه‌بندی، از رنگ جداگانه برای هر خوشه استفاده کرده است. همچنین مرکز این خوشه‌ها که نمونه‌ها بر این اساس تقسیم شده‌اند به صورت علامت X قرمز در نمودار مشخص شده است. کد این بخش در شکل ۲-۴ و رسم نمودار مربوطه در شکل ۲-۵ دیده می‌شود. همان‌طور که به صورت شهودی نیز دیده می‌شود، تعداد داده‌های موجود در دو خوشه تفاوت زیادی با یکدیگر ندارند، در حالی که در Dataset ما تعداد داده با برچسب ۱ از تعداد داده با برچسب ۰ بسیار کمتر است. بنابراین می‌توان پیش‌بینی کرد که خوشه‌بندی ما از کیفیت بالایی برخوردار نیست.

```
plt.figure(figsize=(10, 10))
plt.scatter(samples_pca[:, 0], samples_pca[:, 1], c=kmeans_labels)

centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], marker='x', s=150, linewidths=3, color='r', zorder=10)
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.title("K-means clustering (PCA-reduced data)\nCentroids are marked with red cross")
plt.show()
```

شکل ۲-۴- تصویر کد مربوط به بصری‌سازی نتایج خوشه‌بندی K-means



شکل ۲-۵- نمودار بصری‌سازی نتایج خوشه‌بندی K-means

۲-۳- تحلیل نتایج خوشه‌بندی

برای تحلیل و ارزیابی عملکرد الگوریتم خوشه‌بندی K-means، چندین معیار متفاوت وجود دارد. از آنجا که داده‌های ما دارای برچسب هستند، می‌توان از معیار خلوص[۱] استفاده کرد. این معیار درصد نمونه‌هایی را که به درستی دسته‌بندی شده‌اند را نشان می‌دهد. فرمول آن به صورت زیر است:

$$Purity = \frac{۱}{N}\sum_{i=۱}^{k} max_j(n_{ij})$$

که در آن N تعداد کل نمونه‌ها، k تعداد کل دسته‌ها، n_{ij} تعداد نمونه‌هایی است که همزمان در دسته i و خوشه j قرار دارند و $max_j(n_{ij})$ بیشترین تعداد نمونه‌هایی است که در دسته i و یکی از خوشه‌ها قرار دارند. به عبارت دیگر، ما برای هر دسته، خوشه‌ای را انتخاب می‌کنیم که بیشترین اشتراک با آن دسته را دارد و سپس تعداد نمونه‌های مشترک را جمع می‌کنیم. سپس این مقدار را بر تعداد کل نمونه‌ها تقسیم می‌کنیم تا معیار خلوص را به دست آوریم. این معیار بین صفر و یک متغیر است و هر چه بیشتر باشد نشان‌دهنده‌ی کیفیت بالاتر خوشه‌بندی است. در ادامه نحوه پیاده‌سازی این معیار در کد را توضیح می‌دهیم.

۲-۳-۱- ماتریس پیشایندی

ماتریس پیشایندی یا Contingency Matrix در خوشه‌بندی یک ماتریسی است که نشان می‌دهد که چه تعداد از نمونه‌های داده‌ها که دسته‌های واقعی آن‌ها مشخص است، به چه خوشه‌هایی توسط الگوریتم خوشه بندی تخصیص داده شده‌اند. این ماتریس می‌تواند عملکرد الگوریتم خوشه‌بندی را ارزیابی کند و بگوید که چقدر دقیق است.

برای محاسبه این ماتریس بر روی داده‌های خود، می‌توانیم از کتابخانه sklearn.metrics.cluster استفاده کنیم. در شکل ۲-۶ کد این بخش و در شکل ۲-۷ جزئیات به دست آمده از Contingency Matrix نشان داده شده است.

[۱] Purity Score

```
conting_matrix = cluster.contingency_matrix(labels_true=target_labels, labels_pred=kmeans_labels)
t0, f1, f0, t1 = conting_matrix.ravel()
sum_contin_matrix = np.sum(conting_matrix)
t0_p, f1_p, f0_p, t1_p = ["{:.5f}%".format(item/sum_contin_matrix * 100) for item in [t0, f1, f0, t1]]

print(f"Contingency Matrix: \n{conting_matrix}\n")
print(f"True 0 predicted (Actual = 0 , Predicted = 0): {t0} ({t0_p})")
print(f"False 1 predicted (Actual = 0 , Predicted = 1): {f1} ({f1_p})")
print(f"False 0 predicted (Actual = 1 , Predicted = 0): {f0} ({f0_p})")
print(f"True 1 predicted (Actual = 1 , Predicted = 1): {t1} ({t1_p})")
```

شکل ۲-۶- تصویر کد مربوط به Contingency Matrix و جزئیات آن

```
Contingency Matrix:
[[127959 156356]
 [   173    319]]

True 0 predicted (Actual = 0 , Predicted = 0): 127959 (44.92832%)
False 1 predicted (Actual = 0 , Predicted = 1): 156356 (54.89893%)
False 0 predicted (Actual = 1 , Predicted = 0): 173 (0.06074%)
True 1 predicted (Actual = 1 , Predicted = 1): 319 (0.11201%)
```

شکل ۲-۷- جزئیات به دست آمده از Contingency Matrix

با توجه به نتایج به دست آمده از Contingency Matrix، تعداد ۱۲۷۹۵۹ عدد از نمونه‌ها (حدود ۴۴.۹۲ درصد از کل داده) که برچسب ۰ را داشتند به درستی در خوشه ۰ قرار گرفته‌اند، تعداد ۱۵۶۳۵۶ عدد از نمونه‌ها (حدود ۵۴.۸۹ درصد از کل داده) که برچسب ۰ را داشتند به اشتباه در خوشه ۱ قرار گرفته‌اند، تعداد ۱۷۳ عدد از نمونه‌ها (حدود ۰.۰۶ درصد از کل داده) که برچسب ۱ را داشتند به اشتباه در خوشه ۰ قرار گرفته‌اند و تعداد ۳۱۹ عدد از نمونه‌ها (حدود ۰.۱۱ درصد از کل داده) که برچسب ۱ را داشتند به درستی در خوشه ۱ قرار گرفته‌اند.

۲-۳-۲- مقدار Purity Score

برای محاسبه معیار خلوص یا همان Purity Score، از کلاس ClusteringMetric در کتابخانه permetrics استفاده می‌کنیم. تصویر کد این تابع و خروجی آن در شکل ۲-۸ نشان داده شده است.

```
purity = ClusteringMetric(y_true=target_labels, y_pred=kmeans_labels, decimal = 10).purity_score()
print(f"Clustering Purity: {purity}")
```

```
Clustering Purity: 0.5501093723
```

شکل ۲-۸- محاسبه Purity Score

پس مقدار Purity Score حدودا برابر با ۰.۵۵ شـــد که این نشـــان دهنده آن اســـت که الگوریتم خوشه‌بندی K-means ما عملکرد متوسطی دارد و خیلی خوب عمل نمی‌کند. این امر می‌تواند به دلیل آن باشد که داده‌های ما به صورت نامتوازن و پراکنده توزیع شده‌اند و با وجود کاهش بعد آنها، الگوریتم k-means نمی‌تواند به درســتی تفاوت نمونه‌ها را تنها با داشــتن فاصــله‌ی اقلیدســی آن‌ها با یکدیگر را تشخیص دهد و آن‌ها را با یکدیگر مقایسه کند.

فصل سوم

طبقه‌بندی Logistic Regression

طبقه‌بندی Logistic Regression

در این فصل برای تشخیص تقلب روی کارت‌های اعتباری از روش طبقه‌بندی Logistic Regression استفاده می‌کنیم. برای این کار ابتدا داده را به منظور طبقه‌بندی آماده می‌کنیم. سپس از الگوریتم Logistic Regression استفاده می‌کنیم و دقت و سایر معیارهای مربوط به طبقه‌بندی را ارزیابی می‌کنیم.

۳-۱- آماده‌سازی داده

از آنجا که در قسمت قبل مجموعه داده را بارگیری کردیم و samples و target را تعیین نمودیم، در این قسمت از توضیح مجدد آن خودداری می‌کنیم. برای اینکه بتوانیم از روش طبقه‌بندی استفاده کنیم، باید داده‌های خود را به صورت دو مجموعه آموزش و آزمون تقسیم کنیم. برای مثال ما در اینجا داده‌های خود را به نسبت ۲۰ به ۸۰ به داده‌های آزمون[۱] و آموزش[۲] تبدیل می‌کنیم، یعنی الگوریتم از روی ۸۰ درصد داده‌ها، عملیات یادگیری را انجام می‌دهد و از روی ۲۰ درصد بقیه، خود را ارزیابی می‌کند و نتیجه‌ی ارزیابی را اعلام می‌کند و ما از روی داده‌های ارزیابی می‌توانیم بفهمیم که الگوریتم و مدل ساخته شده توسط آن چقدر دقت داشته است. تبدیل داده‌ها به داده‌های آموزش و آزمون به کمک کلاس train_test_split از کتابخانه sklearn.model_selection انجام می‌شود. کد این بخش در شکل ۳-۱ آورده شده است.

```
x_train, x_test, y_train, y_test = train_test_split(basic_samples, target_labels, test_size=0.2, random_state=10)
```

شکل ۳-۱- تبدیل داده‌ها به داده‌های test و train

حال داده‌های ما آماده هستند تا وارد الگوریتم Logistic Regression شوند.

[۱] Test

[۲] Train

۳-۲- الگوریتم Logistic Regression

برای پیاده‌سازی الگوریتم Logistic Regression از کتابخانه sklearn.linear_model استفاده می‌کنیم. همان‌طور که در شکل ۳-۲ دیده می‌شود، ابتدا یک مدل خالی از Logistic Regression ساخته می‌شود که هنوز آموزش داده نشده است، سپس این مدل با استفاده از داده‌های آموزش که در متغیرهای x_train و y_train ذخیره شده‌اند، آموزش داده می‌شود. x_train شامل ویژگی‌های مستقل و y_train شامل برچسب‌های وابسته است. این خط پارامترهای مدل را با استفاده از روش گرادیان نزولی [۱] بهینه می‌کند و مرز تصمیم‌گیری [۲] را تعیین می‌کند. در نهایت، مدل آموزش دیده برای پیش‌بینی برچسب‌های داده‌های آزمون که در متغیر x_test ذخیره شده‌اند، استفاده می‌شود. x_test شامل ویژگی‌های مستقل است. در اینجا احتمال وقوع هر طبقه را برای هر نمونه محاسبه می‌شود و طبقه‌ای را که بیشترین احتمال را دارد، به عنوان برچسب پیش‌بینی شده انتخاب می‌کند. برچسب‌های پیش‌بینی شده را در متغیر y_predict ذخیره می‌کنیم.

```
logistic_regression_model = LogisticRegression()
logistic_regression_model.fit(x_train, y_train)
y_predict = logistic_regression_model.predict(x_test)
```

شکل ۳-۲- الگوریتم Logistic Regression

۳-۳- ارزیابی عملکرد طبقه‌بندی Logistic Regression

برای ارزیابی عملکرد طبقه‌بندی، چندین معیار مانند Accuracy، Recall، Precision، F1-Score و.. وجود دارد که در ادامه به آن‌ها خواهیم پرداخت.

[۱] Gradient Descent

[۲] Decision Boundary

۳-۳-۱- ماتریس در هم ریختگی

ماتریس در هم ریختگی یا Confusion Matrix یک روش برای خلاصه کردن عملکرد یک مدل یادگیری ماشینی در دسته‌بندی است. ماتریس تعداد پیش‌بینی‌های درست و نادرست مدل را برای هر کلاس نشان می‌دهد. با استفاده از این ماتریس، می‌توانیم معیارهای مختلفی را برای ارزیابی عملکرد مدل محاسبه کنیم.

درایه‌های موجود در این ماتریس به شرح زیر است:

مثبت‌های واقعی (TP): زمانی رخ می‌دهد که مدل یک نقطه داده مثبت را به درستی پیش‌بینی کند.

منفی‌های واقعی (TN): زمانی رخ می‌دهد که مدل یک نقطه داده منفی را به درستی پیش‌بینی کند.

مثبت‌های غلط (FP): زمانی رخ می‌دهد که مدل یک نقطه داده مثبت را به اشتباه پیش‌بینی کند.

منفی‌های غلط (FN): زمانی رخ می‌دهد که مدل یک نقطه داده منفی را به اشتباه پیش‌بینی کند.

برای محاسبه این ماتریس بر روی داده‌های خود، می‌توانیم از کتابخانه sklearn.metrics استفاده کنیم. در شکل ۳-۳ کد این بخش و خروجی آن نشان داده شده است.

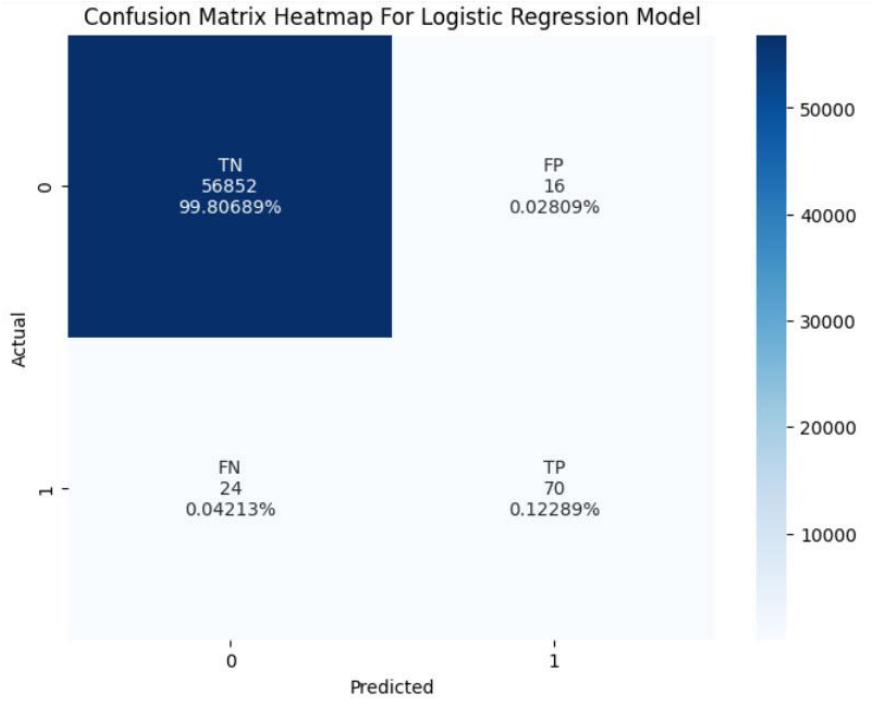
```
conf_matrix = confusion_matrix(y_test, y_predict)
tn, fp, fn, tp = conf_matrix.ravel()
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[56852    16]
 [   24    70]]
```

شکل ۳-۳- محاسبه Confusion Matrix

با توجه به Confusion Matrix به دست آمده، مدل ۵۶۸۵۲ داده بدون تقلب را به درستی به عنوان داده بدون تقلب، ۱۶ داده تقلب را به اشتباه به عنوان داده بدون تقلب، ۲۴ داده بدون تقلب را به اشتباه به عنوان داده تقلب و ۷۰ داده با تقلب را به درستی به عنوان داده تقلب تشخیص داده است. با توجه به این ماتریس، می‌توان حدس زد که طبقه بندی Logistic Regression از عملکرد نسبتا خوبی برخوردار است.

نمودار Heatmap این ماتریس، در شـــکل ۳-۴ نشـــان داده شـــده اســـت. همچنین کد مربوط به این Heatmap که به کمک کتابخانه‌های seaborn و matplotlib.pyplot رســـم شـــده، در شـــکل ۳-۵ آورده شده است.



شکل ۳-۴- نمودار Heatmap برای Confusion Matrix

```
sum_cf_matrix = np.sum(conf_matrix)
tn_p, fp_p, fn_p, tp_p = ["{:.5f}%".format(item/sum_cf_matrix * 100) for item in [tn, fp, fn, tp]]
cf_matrix_details = np.array([[f"TN\n{tn}\n{tn_p}", f"FP\n{fp}\n{fp_p}"], [f"FN\n{fn}\n{fn_p}", f"TP\n{tp}\n{tp_p}"]])

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=cf_matrix_details, fmt="", cmap="Blues", xticklabels=["0", "1"], yticklabels=["0", "1"])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap For Logistic Regression Model')
plt.show()
```

شکل ۳-۵- تصویر کد رسم Heatmap

۳-۳-۲- معیارهای ارزیابی طبقه‌بندی

همان‌طور که گفته شـــد، علاوه بر معیار دقت (Accuracy)، معیارهای دیگری نیز برای ارزیابی الگوریتم طبقه‌بندی وجود دارند. معیارهای ارزیابی مدل طبقه‌بندی نشـان می‌دهند که چقدر مدل ما قادر اسـت به داده‌های جدید که در مجموعه داده‌های آموزشـی نیسـتند، به درسـتی برچسـب بزند. ما در اینجا از چهار

معیار Accuracy، Recall، Precision و F1-Score برای ارزیابی کیفیت طبقه‌بندی Logistic Regression استفاده می‌کنیم که نحوه‌ی محاسبه این معیارها و کد مربوط به آن‌ها در شکل ۳-۶ و خروجی این کد که مقدار این معیارها برای مدل ما را مشخص می‌کند در شکل ۳-۷ آورده شده است.

```
accuracy = (tp + tn) / (tp + tn + fn + fp)
recall = tp / (tp + fn)
precision = tp / (tp + fp)
f1_score = (2 * precision * recall)/(precision + recall)

print(f"Accuracy: {(accuracy)}")
print(f"Recall: {recall}")
print(f"Precision: {precision}")
print(f"F1-Score: {f1_score}")
```

شکل ۳-۶- تصویر کد نحوه محاسبه معیارهای ارزیابی طبقه‌بندی

```
Accuracy: 0.9992977774656788
Recall: 0.7446808510638298
Precision: 0.8139534883720930
F1-Score: 0.7777777777777778
```

شکل ۳-۷- مقدار معیارهای ارزیابی طبقه‌بندی Logistic Regression

Accuracy نسبت تعداد پیش‌بینی‌های صحیح به تعداد کل پیش‌بینی‌ها است. این معیار نشان می‌دهد که چقدر مدل در کل دقیق است. با توجه به نتایج به دست آمده، مقدار Accuracy مدل ما بسیار بالا است (حدود ۰.۹۹۹) که نشان می‌دهد الگوریتم طبقه‌بندی Logistic Regression تقریبا تمام داده‌های آزمون را به درستی طبقه بندی کرده است.

Recall نسبت تعداد پیش‌بینی‌های صحیح مثبت به تعداد کل داده‌های مثبت است. این معیار نشان می‌دهد که چقدر مدل قادر است داده‌های مثبت را تشخیص دهد. مقدار Recall مدل ما متوسط است (حدود ۰.۷۴۴) که نشان می‌دهد مدل برخی از داده‌های مثبت را از دست داده است، یعنی مدل برخی از داده‌های بدون تقلب را به عنوان تقلب تشخیص داده است.

Precision نسبت تعداد پیش‌بینی‌های صحیح مثبت به تعداد کل پیش‌بینی‌های مثبت است. این معیار نشـــان می‌دهد که چقدر پیش‌بینی‌های مثبت مدل قابل اعتماد هســـتند. مقدار Precision مدل ما بالا اسـت (حدود ۰.۸۱۳) که نشـان می‌دهد مدل بیشـتر پیش‌بینی‌های مثبت را به درسـتی انجام داده اسـت. یعنی از بین تمام داده‌هایی که مدل به عنوان تقلب تشـخیص داده اسـت، بیش‌تر آن‌ها واقعا داده‌ی تقلب بوده‌اند.

F1-Score میانگین هارمونیک Recall و Precision اســـت. این معیار نشـــان می‌دهد که چقدر مدل توازن بین Recall و Precision دارد. مقدار F1-Score مدل ما متوسط رو به بالا است (حدود ۰.۷۷۷) که نشان می‌دهد مدل ما در هر دو معیار بهترین عملکرد را نداشته است.

در کل، می‌توان گفت که عملکرد مدل ما در معیار Accuracy بسیار خوب، در معیار Precision خوب، در معیار F1-Score و معیار Recall متوسـط بوده اسـت. بنابراین، مدل ما می‌تواند بهبود یابد اگر بتواند داده‌های تقلب را بهتر تشخیص و Recall را افزایش دهد.

فصل چهارم

لینک کد گزارش

لینک کد گزارش

کد تشخیص تقلب مالی به دو روش خوشه‌بندی K-means و طبقه‌بندی Logistic Regression در Google Colab نوشته شده و لینک آن در زیر قابل دسترسی است:

لینک کد گزارش ۵ در Google Colab

فصل پنجم

جمع‌بندی و نتیجه‌گیری

جمع‌بندی و نتیجه‌گیری

با توجه به نتایج به دســـت آمده از دو روش K-means و Logistic Regression، می‌توان به راحتی فهمید که برای چنین داده‌ای که نمونه‌های تقلب و بدون تقلب به درســتی توزیع نشــده‌اند و توازن خوبی ندارند، الگوریتم طبقه‌بندی Logistic Regression که از روش‌های یادگیری تحت نظارت است، نسبت به الگوریتم خوشه‌بندی K-means که از روش‌های یادگیری بدون نظارت است، بسیار بهتر عمل می‌کند و پیش‌بینی بهتری در تشـــخیص داده‌های تقلب از دیگر داده‌های تراکنش مالی اســـت. اما با این وجود، مدلی که از الگوریتم طبقه‌بندی Logistic Regression ســـاختیم، قابلیت بهبود را دارد، زیرا به جز مقدار Accuracy، مقدار معیارهای Recall، Precision و F1-Score آنقدر بالا نیســـت که بتوانیم با اطمینان خاطر بگوییم که الگوریتم بهترین عملکرد خود را دارد و بیش از این نمی‌تواند بهبود یابد.

منابع

"Faradars," [Online]. Available: https://b.fdrs.ir/15.

"Faradars," [Online]. Available: https://b.fdrs.ir/2n2.

"Kaggle," [Online]. Available: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.

"Shabakeh," [Online]. Available: https://shabakeh-mag.com/node/20942.

"Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Principal_component_analysis.

"Scikit-Learn," [Online]. Available: https://scikit-learn.org/stable/.

"Geeks," [Online]. Available: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/.

"Hooshio," [Online]. Available: https://hooshio.com/%D8%B7%D8%A8%D9%82%D9%87-%D8%A8%D9%86%D8%AF%DB%8C/