گزارش پروژه ی اول

یافتن متغییرهای مربوط به نمره ی ریاضی دانش آموزان کلاس هشتم

فاطمه کریمی(۴۰۰۷۲۳۱۳۱)

بیستم اردیبهشت ۱۴۰۱

چکیده

در این پروژه قصد داریم با در دست داشتن اطلاعات حدود ۵۸۰۰ دانش آموز، عوامل موثر بر نمره ی ریاضی آنان را بیابیم. برای این منظور، نمره های این دانش آموزان به همراه پاسخ آنان به شرایط محیطی آنان(مانند جنسیت، میزان تحصیلات پدر و مادر، سابقه ی کاری آموزگار و ...) به ما داده شده است. برای پاسخ به این پرسش، پس از تمیز کردن داده های ورودی، با حساب کردن میزان همبستگی بین هریک از متغییرها با متغییر خروجی نمره، موثر ترین عوامل بر نمره ی دانش آموز را می یابیم.

۱ راه اندازی

برای انجام این پروژه، از زبان برنامه نویسی پایتون و کتابخانه های مربوط به پردازش داده مانند pandas، و یادگیری ماشین مانند scipy و sklearn استفاده می کنیم. برای کشیدن نمودارها، کتابخانه ی matplotlib به کار رفته است. پروژه ی پیاده سازی شده در Github می باشد.

۲ شناسایی داده ها

گام نخست، شناسایی داده ها داده شده است. اطلاعات مربوط به هر متغییر در فایلی به نام T2019_G8_Codebook.xlsx آمده است. در این فایل به ازای هر متغییر، نام آن، سطح آن، بازه ای که مقادیر متغییر و جزییات آن آمده است. متغییرهای ورودی در سه نوع Ordinal، Nominal و Scale تعریف شده اند. برای متغییرهای دسته ای، توضیحات بیانگر گزینه های موجود برای انتخاب بوده است. مثلا برای متغییر جنسیت، در توضیحات عبارت 1.Girl 2.Boy آمده است که بیان می کند برای دانش آموزان دختر در متغییر جنسیت، عدد ۱ و برای دانش آموزان پسر عدد ۲ قرار داده شده است. مورد دیگر، مقدارهای بیشینه و کمینه برای متغییرهاست که تنها برای برخی این مقدارها قرار داده شده است. برای مثال، برای متغییر سن دانش آموز بازه ی ۹ تا ۱۹ آمده است که بدین معناست هر عدد خارج از این بازه یک مقدار نامعتبر خواهد بود.

برای دسترسی به اطلاعات متغییرها، در پیاده سازی کلاسی با نام Codebook تعریف شده است. این کلاس اطلاعات مربوط به ویژگی ها را از فایل اکسل مربوط به آن بیرون می کشد و برای هر ویژگی، آبجکتی از کلاس Attribute می سازد و اطلاعات مربوط به ویژگی را در این آبجکت ذخیره می کند. به کمک کلاس Codebook، می توان به فهرستی از همه ی ویژگی ها دسترسی داشت. در دست داشتن این اطلاعات، به ما در پیش پردازش داده ها کمک می کند.

۳ پیش پردازش داده ها

پیش پردازش داده ها گام باارزشی در پردازش داده ها می باشد. این فاز از سه بخش تشکیل می شود: یکم پر کردن مقدارهای null و داده های پرت، دوم یافتن متغییرهای تکراری با حساب کردن همبستگی بین داده ها و سوم جایگزین کردن نمره های دانش آموزان با اعداد. در ادامه هر بخش به طور جداگانه توضیح داده می شود.

۳-۱ پرکردن داده های خالی و داده های پرت

برای جایگزین کردن مقدارهای null برای یک ویژگی، ابتدا تعداد مقدارهای null را می شماریم. در صورتی که نسبت تعداد مقادیر null به تعداد کل مقادیر از ۵۰٪ بیشتر باشد، آنگاه کل آن متغییر را از داده های ورودی حذف می کنیم. علت این کار آنست که اگر درصد زیادی از داده ها null باشد، آنگاه می بایست این مقادیر خالی را با داده ای مانند میانه پر کنیم که این کار غیر منطقی ست؛ چرا که در واقع میانه ی کمتر از ۵۰٪ داده ها را برای آنان قرار می دهیم. بنابراین، در چنین شرایطی کل متغییر را از داده های ورودی حذف می کنیم. برای ویژگی هایی که میزان داده های null آنها از مقدار آستانه کمتر است، بنابر نوع ویژگی، چگونگی پرکردن آنها را بیان می کنیم.

پر کردن داده های null برای متغییرهای از نوع Nominal

همانگونه که می دانید، متغییر های nominal از نوع متغییر های دسته ای هستند و تفاوت آنها با متغییر های ordinal آنست که هیچ دسته نسبت به دسته ی دیگر برتری ندارد. این موضوع باعث می شود که برای این دسته از متغییرها، میانگین و میانه بی معنی باشد. بنابراین ما از مقدار مد(mod) برای پر کردن مقادیر null استفاده می کنیم. مد داده ایست که بیشترین تکرار را در ورودی داشته است. گاهی در ورودی، مقدار unknown به جای آنکه با null پر شود، با مقدارهای عددی بسیار بزرگ مانند ۹۹۹۹ پر شده است. برای متغییرهای nominal، می توان بازه ی مجاز را از روی توضیحات آنها که در فایل Codebook آمده است، به دست آورد. سپس، همه ی مقادیر غیر مجاز را با مد پر کرد.

شایان ذکر است که پر کردن مقادیر غیر مجاز و null با مد، در صورتی انجام می شود که نسبت این داده ها به کل داده ها کمتر از ۰.۱ باشد. در صورتی که بیشتر بود، گزینه ی جدیدی به مجموعه گزینه های متغییر با نام UNKNOWN اضافه می شود و برای این دسته از داده ها مقدار (بیشترین شماره دسته ی موجود + ۱) را قرار می دهیم. متغییرهای BCBG10B، BCBG21B، BCBG21C، BSBM27AA و BSDG05S متغییرهایی هستند که به آنها گزینه ی UNKKNOWN اضافه شده است.(قطعه کد ۱)

```python
class NominalPreprocess(LevelPreprocess):
    def fill_missing_value(self, df, attr_list):
        result_df = df
        for attr in attr_list:
            result_df = self.fill_missing_value_by_col(result_df, attr)
        return result_df

    def fill_missing_value_by_col(self, df, attr):
        if attr.variable not in df.columns:
            self.log_key_not_exist(attr)
            return df

        pd.to_numeric(df[attr.variable])

        self.replace_out_range_values(df, attr, np.nan)
        if self.is_attr_too_null(df, attr):
            df = df.drop(labels=[attr.variable], axis=1)
            return df

        mod = self.get_mod(df, attr)
        if df[attr.variable].isna().sum() > INTRODUCE_OPTION_RATIO * len(df.index):
            options = attr.get_options()
            options.append('UNKNOWN')
            attr.set_options(options)
            df[attr.variable] = df[attr.variable].fillna(len(options))
            self.log_option_added(attr, 'UNKNOWN')
        else:
            df[attr.variable] = df[attr.variable].fillna(mod)
        return df
```

قطعه کد 1

پر کردن داده های null برای متغییرهای از نوع Ordinal

از آنجایی که در متغییرهای Ordinal ترتیب معنادار است، برای جایگزین کردن مقادیر null و غیر مجاز(خارج از بازه) از میانه استفاده می کنیم. نکته ی قابل توجه آنست که برای متغییرهای Ordinal امکان اضافه کردن دسته ی جدید وجود ندارد؛ بنابراین همه ی مقادیر null با داده ی میانه پر می شوند.

```python
class OrdinalPreprocess(LevelPreprocess):
    def fill_missing_value(self, df, attr_list):
        result_df = df
        for attr in attr_list:
            result_df = self.fill_missing_value_for_col(result_df, attr)
        return result_df

    def fill_missing_value_for_col(self, df, attr):
        pd.to_numeric(df[attr.variable])

        if self.is_attr_too_null(df, attr):
            df = df.drop(labels=[attr.variable], axis=1)
            self.log_attr_removed(attr)
        else:
            med = self.get_data_median_for(df, attr)
            self.replace_invalid_values(df, attr, med)

        return df
```

قطعه کد 2

پر کردن داده های null برای متغییرهای از نوع Scale

برای متغییرهای از نوع scale، ابتدا می بایست داده های پرت را بیابیم. برای یافتن داده های پرت، از روش whisker box اسفاده می کنیم. بدین شکل که داده هایی که خارج از بازه ی [q1 – 1.5 * iqr, q3 + 1.5 * iqr] قرار دارند را داده ی پرت در نظر می گیریم و این داده ها را با مقدار null جایگزین می کنیم. حال، درصد داده های null به کل داده ها را در نظر می گیریم؛ اگر بیش از ۵۰٪ بود، کل متغییر را حذف می کنیم. در غیر این صورت، متغییر را نگه می داریم. اگر متغییر حذف نشد، توزیع آن را بررسی می کنیم. اگر از توزیع نرمال پیروی می کرد، مقادیر null را با میانگین و در غیر این صورت با میانه پر می کنیم. برای بررسی توزیع داده ی متغییر، از کتابخانه ی scipy بهره می بریم. این کتابخانه تابعی به نام normaltest دارد و یکی از مقادیری که این تابع بر می گرداند مقدار p-value است. اگر مقدار p-value کمتر از ۰.۰۵ بود، آنگاه توزیع را نرمال در نظر می گیریم. البته هیچ یک از متغییرهای این مساله توزیع نرمال نداشتند و بنابراین درنهایت مقادیر null برای آنها با میانه پر شد.(قطعه کد ۳)

```
class OrdinalPreprocess(LevelPreprocess):
    def fill_missing_value(self, df, attr_list):
        result_df = df
        for attr in attr_list:
            result_df = self.fill_missing_value_for_col(result_df, attr)
        return result_df

    def fill_missing_value_for_col(self, df, attr):
        pd.to_numeric(df[attr.variable])

        if self.is_attr_too_null(df, attr):
            df = df.drop(labels=[attr.variable], axis=1)
            self.log_attr_removed(attr)
        else:
            med = self.get_data_median_for(df, attr)
            self.replace_invalid_values(df, attr, med)

        return df
```

قطعه کد 3

۳-۲ حذف متغییرهای تکراری

پس از پر کردن متغییرهای خالی، نوبت به حذف متغییرهای تکراری ست. برای انجام این کار، مقدار همبستگی را به صورت دو به دو برای متغییرهای هر نوع حساب می کنیم. سپس، آنهایی که قدرمطلق همبستگی آنها بیشتر از ۰.۸ است را در نظر می گیریم از بین آن دو، یکی را حذف می کنیم. از آنجایی که روش محاسبه ی همبستگی برای هر نوع متفاوت است، در ادامه جزییات این کار را بیان می داریم.

محاسبه ی همبستگی بین دو متغییر از نوع scale

برای محاسبه ی همبستگی بین متغییر scale، از روش pearson استفاده می کنیم. برای این منظور، از تابع corr در کتابخانه ی pandas بهره می بریم. سپس، ماتریس همبستگی حاصل را به کمک کتابخانه ی matplotlib در قالب heatmap نمایش می دهیم. شکل ۱، نمودار همبستگی را نشان می دهد. همانگونه که از نمودار پیداست، هیچ کدام از دو متغییر همبستگی بسیار قوی وجود ندارد؛ بنابراین هیچ یک از متغییرهای از نوع scale به دلیل همبستگی حذف نمی شوند.

```
def get_correlation_matrix(self, df, attr_list):
    scale_columns = [
        attr.variable for attr in attr_list if attr.variable in df.columns]

    scale_df = df[scale_columns]
    return scale_df.corr(method='pearson')
```

قطعه کد 4- استفاده از متد pearson برای محاسبه همبستگی

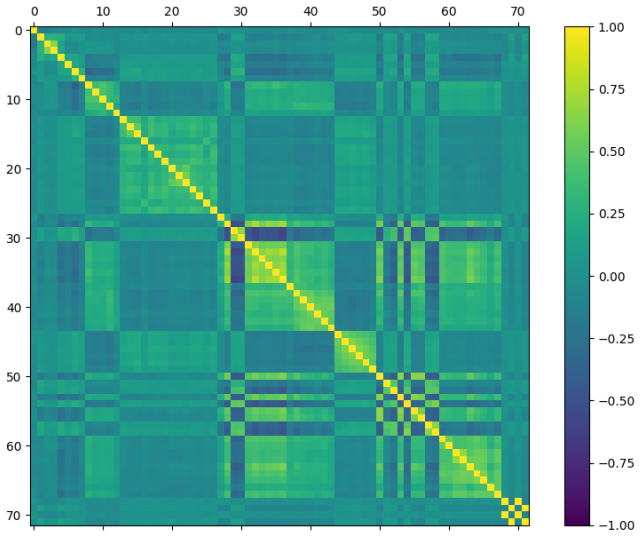
شکل 1- نمودار همبستگی میان دو متغییر از نوع scale

محاسبه همبستگی بین دو متغییر از نوع Ordinal

برای محاسبه ی میزان همبستگی بین دو متغییر ordinal، از روش spearman rank correlation استفاده می کنیم. همانند قسمت قبل، تابع corr از کتابخانه ی pandas را فرا می خوانیم و روش محاسبه ی همبستگی را spearman تعیین می کنیم. شکل ۲، نمودار heatmap را نشان می دهد.

```
def get_correlation_matrix(self, df, attr_list):
    ordinal_columns = [
        attr.variable for attr in attr_list if attr.variable in df.columns]
    ordinal_df = df[ordinal_columns]
    return ordinal_df.corr(method='spearman')
```

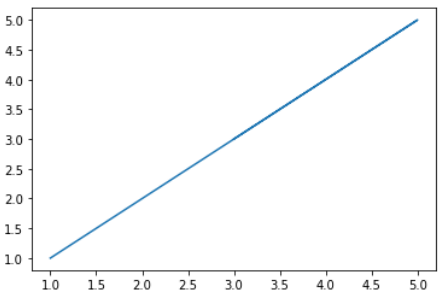
قطعه کد 5- محاسبه همبستگی میان دو متغییر Ordinal



شکل 2- نمودار همبستگی میان دو متغییر از نوع ordinal

پس از محاسبه ی ماتریس همبستگی، سه ویژگی BSBM42AA،BSBM42BA و BSDGSLM به دلیل همبستگی قوی با متغییرهای دیگر حذف می شوند. برای نمونه، متغییر BSBM42AA با متغییر BSBM26AA همبسته است و نمودار آنها به شکل زیر است:

```
In [8]: plt.plot(df['BSBM42AA'], df['BSBM26AA'])
Out[8]: [<matplotlib.lines.Line2D at 0x7f1eb905a3d0>]
```



شکل 3- همبستگی دو متغییر BSBM42AA و BSBM26AA

محاسبه ی همبستگی بین دو متغییر از نوع Nominal

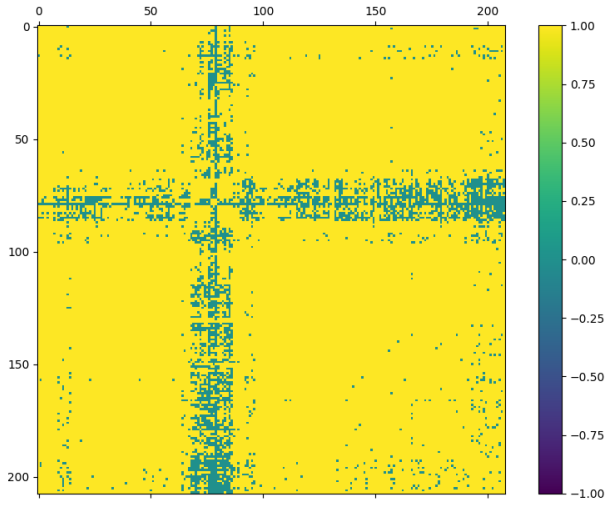
برای محاسبه ی همبستگی بین دو متغییر nominal از chi-square استفاده می کنیم. برای محاسبه ی chi-square، از تابع chi2_contingency درکتابخانه ی scipy.stats کمک می گیریم. یکی از مقادیر برگشتی این تابع مقدار p_value است. اگر p_value کمتر از ۰.۰۵ باشد یعنی دو متغییر همبسته هستند. نمودار heatmap در شکل ۴ آمده است.

با توجه به نمودار، به نظر می رسد درصد زیادی از متغییرها با یکدیگر همبستگی دارند. به همین دلیل، تصمیم بر آن شد تا متغییرهای nominal همبسته حذف نشوند(!).

```
def calc_chi_square(self, col1, col2):
    contingency = pd.crosstab(col1, col2)
    stat, p, dof, expected = chi2_contingency(contingency)
    return p <= 0.05

def get_correlation_matrix(self, df, attr_list):
    nominal_columns = [
        attr.variable for attr in attr_list if attr.variable in df.columns]
    nominal_df = df[nominal_columns]
    return nominal_df.corr(method=self.calc_chi_square)
```

قطعه کد 6- محاسبه همبستگی میان دو متغییر nominal



شکل 4-نمودار همبستگی میان دو متغییر nominal

۳-۳ جایگزین کردن نمره ی دانش آموزان با مقادیر عددی

در پایان فرایند پاکسازی داده ها، متغییرهای نمره شامل finalscore، finalscorealgebra، finalscoredat، finalscoregeo و finalscorenum را با نمره های عددی جایگزین می کنیم. برای این کار، نمره ی A را با ۵، نمره ی B را با ۴، نمره ی C را با ۳، نمره ی D را با ۲ و نمره ی E را با ۱ جایگزین می کنیم. در آخر نمره های null را با میانه پر می کنیم.

٤ گزینش متغییرهای موثر بر نمره ی دانش آموزان

برای گزینش متغییرهای موثر بر نمره ی دانش آموزان، از کتابخانه ی sklearn.feature_selection استفاده می کنیم. گام نخست آنست که ستون هایی که یک مقدار ثابت دارند(واریانس داده های آنها صفر است) را حذف کنیم. حذف نکردن این متغییرها موجب دریافت هشدارهایی از طرف توابع این کتابخانه می شود. در گام دوم به کمک تابع SelectKBest، k تا از موثرترین متغییرها بر نمره را انتخاب می کنیم. در اینجا مقدار k را برابر یک سوم تعداد متغییرها قرار می دهیم. تابع SelectKBest، متدی را برای انتخاب ویژگی ها به عنوان پارامتر دریافت می کند. برای متغییرهای categorical(ordinal و nominal) این متد chi2 و برای متغییرهای scale این متد ANOVA(که در کتابخانه ی sklearn.feature_selection، f_classif نام دارد) می باشد. کار این تابع آنست که میزان همبستگی میان متغییرهای ورودی و متغییر خروجی(در اینجا نمره دانش آموزان) را محاسبه کند و سپس K تا از متغییرهایی که بیشترین همبستگی را داشتند، گزارش کند.

```
def find_top_scale_features(df, attr_list, target):
    scale_colnames = [attr.variable for attr in attr_list
                      if attr.variable in df.columns
                          and (attr.level == dataConst.AttributeLevel.SCALE)]
    x = df[scale_colnames]
    y = df[target]
    x = remove_constants(x, attr_list)

    selector = SelectKBest(f_classif, k=len(scale_colnames) // 3)
    x_new = selector.fit_transform(x, y)
    cols = selector.get_support(indices=True)
    top_scale_features = x.iloc[:, cols]
    top_attributes = []
    for attr_name in top_scale_features.columns:
        top_attributes.append(attr_name)
    return top_attributes
```

قطعه کد 7- انتخاب متغییرهای scale با بیشترین همبستگی با متغییر هدف

۵ نتایج

نتایج مربوط به پیاده سازی با فرمت JSON در پوشه ی docs/results/ به تفکیک سوال آمده است.

٦ منابع

[1] Json Browlee, 2020, *How to choose a feature selection method for machine learning*, Access 2 April 2022, https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

[2]Json Browlee, 2020, *Feature selection for machine learning in python*, Access 2 April 2022, https://machinelearningmastery.com/feature-selection-machine-learning-python/