به نام خدا

دانشگاه علامه طباطبایی

درس روش‌های چندمتغیره پیوسته ۲

گزارش پروژه های ارائه شده تا مقطع میانترم

استاد: دکتر وحید رضایی تبار

پژوهشگر: علی شکارچی

سال تحصیلی: نیمسال اول ۱۴۰۳

فهرست

۱ پروژه ۱ - برازش رابطه رگرسیون،بررسی پیش فرض‌ها

۲ پروژه ۲ - برازش رگرسیون مولفه‌های اصلی بر مجموعه داده‌های داری هم‌خطی معنی‌دار

۳ پروژه ۳ - نگاشت ماتریسی ضرایب مدل رگرسیون مولفه‌های اصلی به ضرایب مناسب برای متغیرهای مستقل

۴ پروژه ۴ - برازش رگرسیون های ریج (ستیغی) و لاسو

۵ پروژه ۵ – تحلیل عاملی

پروژه ۱

برازش رابطه رگرسیونی، بررسی پیش‌فرض‌ها با روش‌های آماری شامل نرمال بودن خطا، ثبات واریانس، هم خطی بر روی مجموعه داده‌های مناسب

مجموعه داده‌های Combined Cycle Power Plant (نیروگاه سیکل ترکیبی) از پایگاه UC Irvine Machine Learning Repository برای پروژه اختیار گردیده است.

این دیتاست شامل ۹,۵۶۸ داده است که طی ۶ سال (۲۰۰۶-۲۰۱۱) جمع آوری شده و مربوط به نیروگاه برق سیکل ترکیبی می‌باشد. متغیر وابسته ما در اینجا (PE) که انرژی برق خروجی نیروگاه است و متغیرهای مستقل شامل (AT) دما ، (V) خلاء اگزوز ، (AP) فشار و (RH) که رطوبت هوا است می‌شود.

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
data = pd.read_excel('Folds5x2_pp.xlsx')
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
print("Feautures\n", X.head())
print("Shape of Features Dataframe:", X.shape)
print("Targets\n", y.head())
print("Shape of Targets:", y.shape)
```

```
Feautures
       AT      V       AP     RH
0  14.96  41.76  1024.07  73.17
1  25.18  62.96  1020.04  59.08
2   5.11  39.40  1012.16  92.14
3  20.86  57.32  1010.24  76.64
4  10.82  37.50  1009.23  96.62
Shape of Features Dataframe: (9568, 4)
Targets
 0    463.26
1    444.37
2    488.56
3    446.48
4    473.90
Name: PE, dtype: float64
Shape of Targets: (9568,)
```

این دیتاست در ۹,۵۶۸ ردیف و ۵ ستون ارائه شده است.

از ستون اول تا چهارم داده‌ها را که شامل (AT),(V),(AP),(RH) است در متغیر x ریخته که به عنوان متغیرهای مستقل می‌باشند.

ستون پنجم داده‌ها را که مربوط به (PE) است به عنوان متغیر پاسخ(وابسته) مدنظر قرار داده و در متغیر y ریخته.

سپس ۵ سطر اول متغیرهای x, y نمایش داده می‌شود.

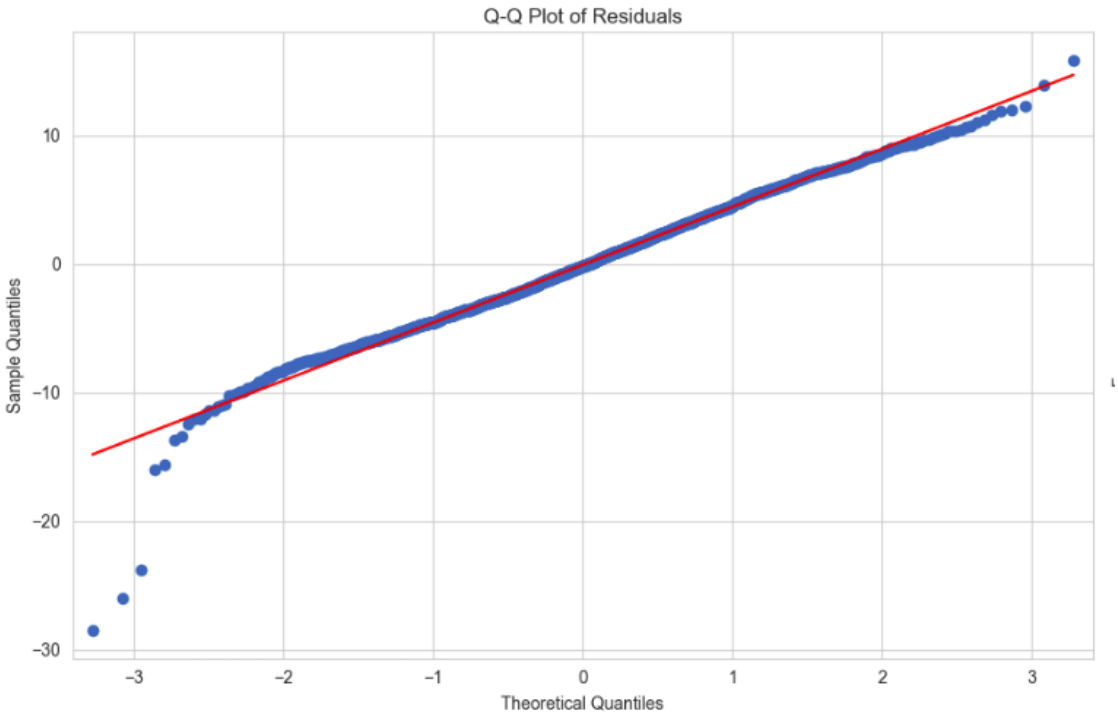
[244]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

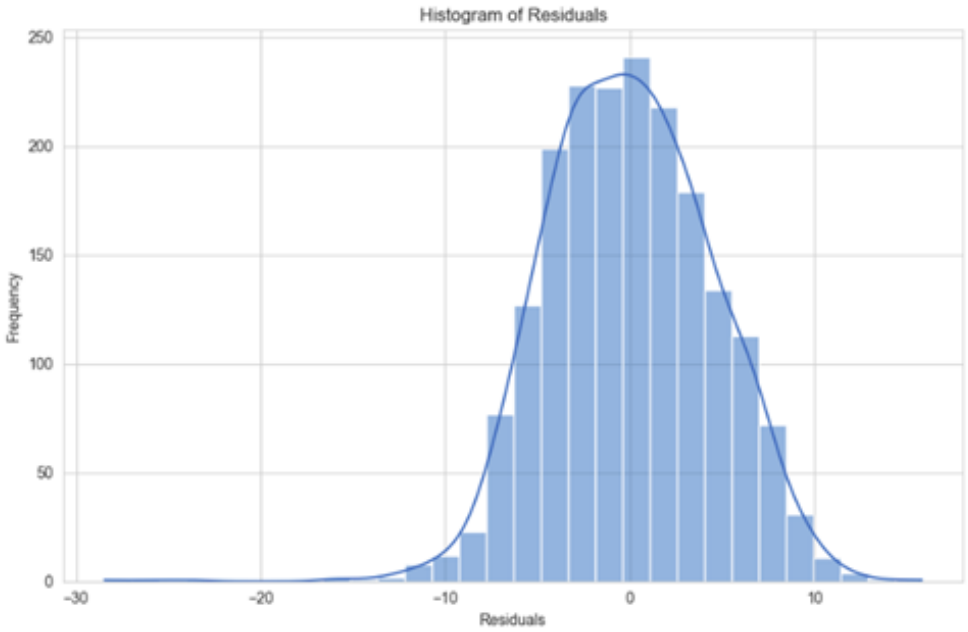
[245]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

model = LinearRegression()
model.fit(X_scaled, y)
y_pred = model.predict(X_scaled)
print(f"Intercept: {model.intercept_:.3f}")
Co = [round(i, 3) for i in model.coef_]
print(f"Coefficients: {Co}")
r2 = r2_score(y, y_pred)
print(f"R^2 score: {r2:.3f}")
```

برای استاندارد سازی با استفاده از تابع Standard Scaler این کار را انجام می‌پذیرد. و سپس یک مدل رگرسیون خطی با کمک تابع LinearRegression فراخوانی شده از پکیج sklearn پیاده کرده و به کمک تابع model.fit ضرایب رگرسیونی روی داده‌های آموزش بدست می‌آید و مقدار R2 آنرا محاسبه می‌شود، حال مقدار بتا صفر و ضرایب رگرسیونی و R2 نمایش داده می‌شود.

در ادامه به بررسی تایید یا عدم تایید فرض نرمال بودن داده‌ها پرداخته می‌شود.

[8]:
```
import statsmodels.api as sm
residuals = y_test - y_pred

plt.figure(figsize=(18, 6))

plt.subplot(1, 2, 1)
sm.qqplot(residuals, line='s', ax=plt.gca())
plt.title("Q-Q Plot of Residuals")

plt.subplot(1, 2, 2)
sns.histplot(residuals, bins=30, kde=True, ax=plt.gca())
plt.title('Histogram of Residuals')
plt.xlabel('Residuals')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```





در این قسمت نمودارهای Q_Q plot و هیستوگرام مانده‌ها را رسم کرده و به وضوح فرض نرمال بودن مانده‌ها تایید می‌شود.

در ادامه به بررسی و تایید این مطلب به کمک آزمون کلموگروف-اسمیرنوف پرداخته می‌شود.

[248]:
```
from scipy import stats

mean_residuals = np.mean(residuals)
std_residuals = np.std(residuals)
ks_statistic, p_value = stats.kstest(residuals, 'norm', args=(mean_residuals, std_residuals))
print(f'KS Statistic: {ks_statistic:.5f}')
print(f'p-value: {p_value:.5f}')
```

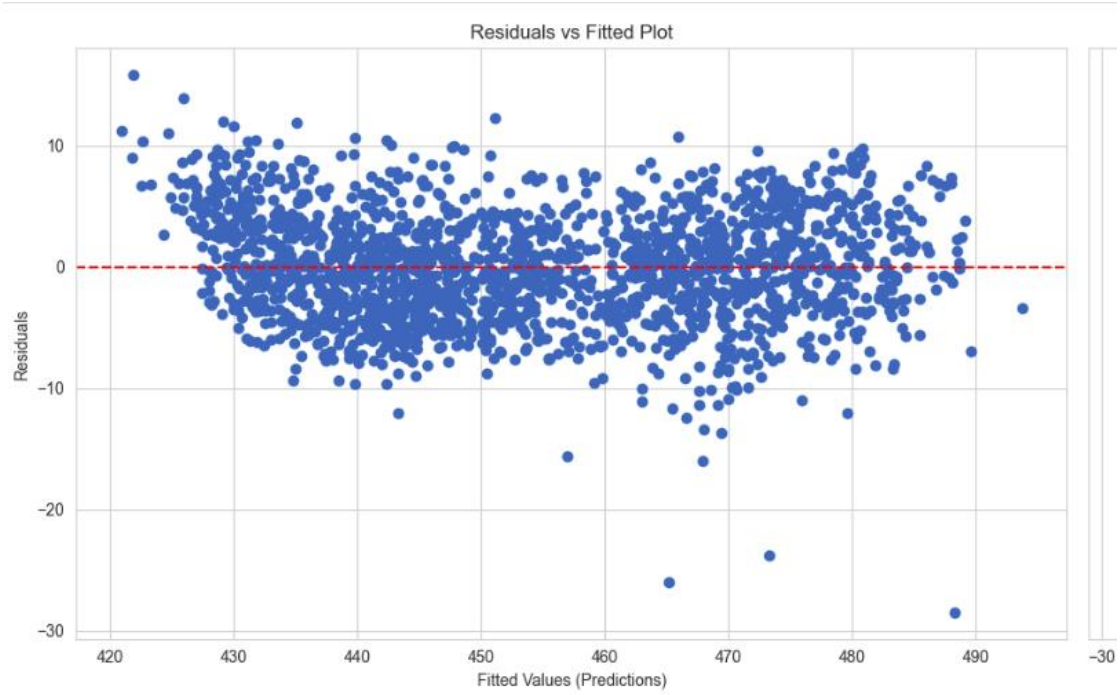
```
KS Statistic: 0.02096
p-value: 0.00044
```

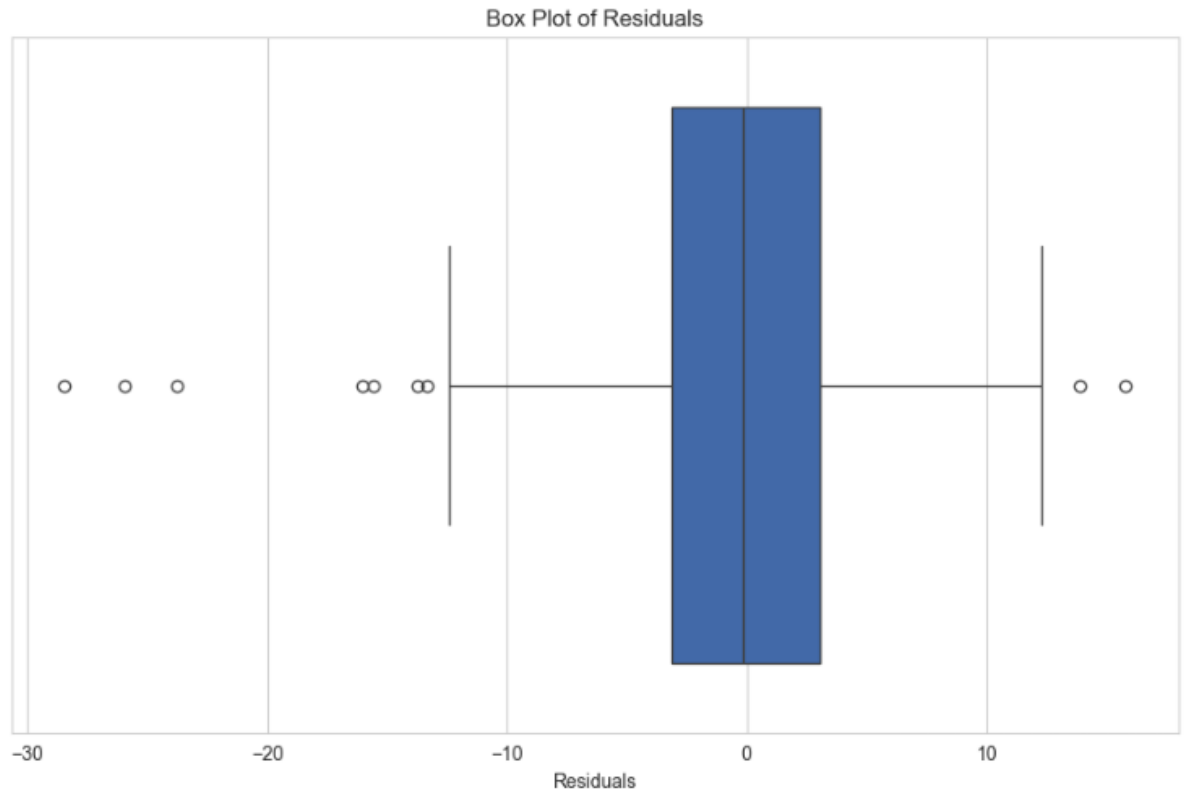
مقدار آماره KS: اگر مقدار این آماره نزدیک به صفر باشد فرض نرمالیتی تایید و اگر نزدیک به یک باشد فرض نرمالیتی رد می‌شود که در اینجا با توجه به اینکه مقدار آن ۰.۰۲ شده فرض نرمالیتی داده‌ها تایید می‌شود.

مقدار p_value: با توجه به مقدار این آماره که برابر ۰.۴ شده و مقدار بالایی است فرض نرمالیتی رد نمی‌شود زیرا اگر مقدار پی ولیو زیر ۰.۰۵ باشد فرض نرمالیتی را رد می‌کنیم و در غیر این صورت رد نمی‌کنیم.

در بخش بعد به بررسی تایید یا عدم تایید فرض ثبات واریانس پرداخته می‌شود.

```
plt.figure(figsize=(18, 6))

plt.subplot(1, 2, 1)
plt.scatter(y_pred, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Fitted Values (Predictions)')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Plot')

plt.subplot(1, 2, 2)
sns.boxplot(x=residuals)
plt.title('Box Plot of Residuals')
plt.xlabel('Residuals')

plt.tight_layout()
plt.show()
```


Residuals vs Fitted Plot
Residuals
Fitted Values (Predictions)
10
0
−10
−20
−30
420
430
440
450
460
470
480
490

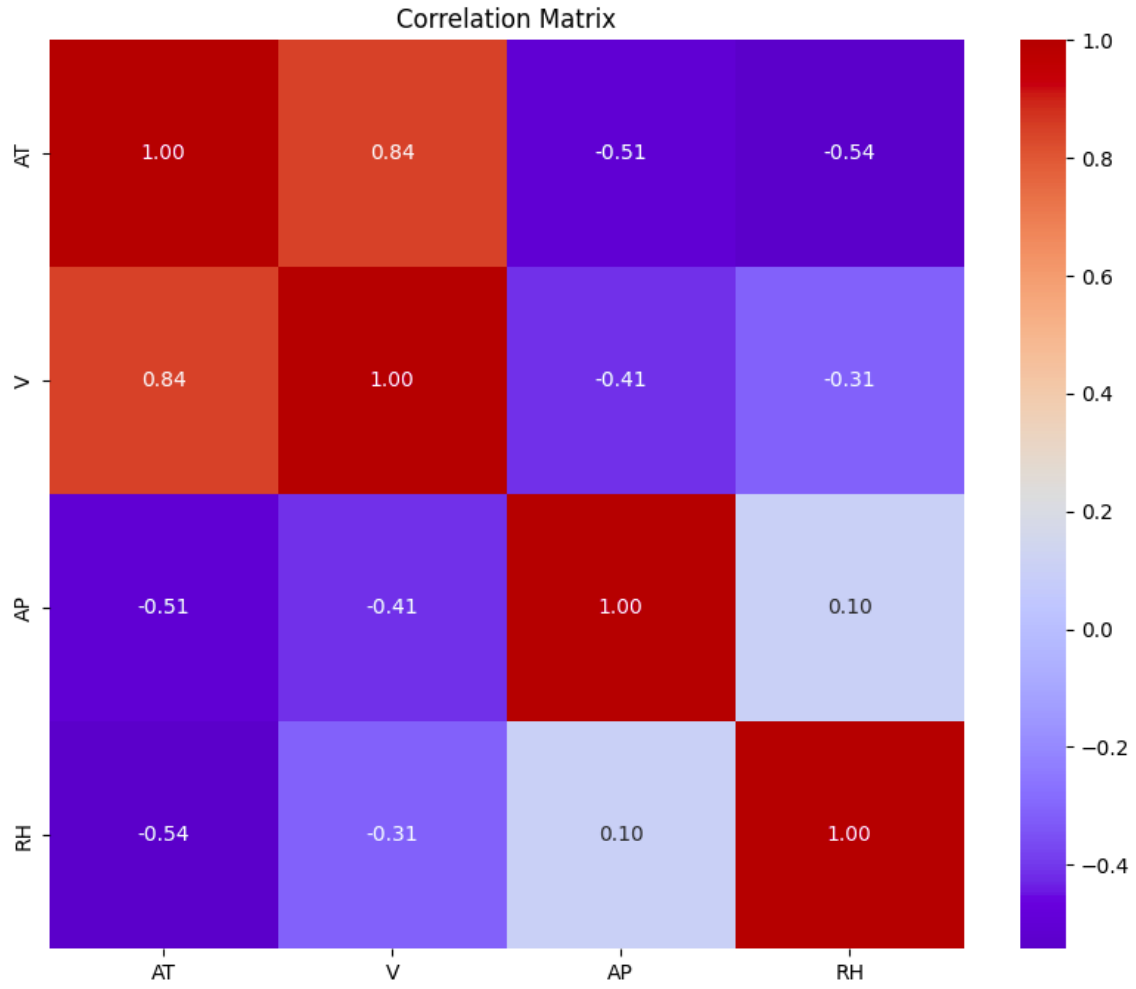

Box Plot of Residuals
Residuals
−30
−20
−10
0
10

نمودارهای مانده‌ها در برابر پیش‌بینی و باکس پلات رسم می‌شوند.

از این نمودارها می‌توانیم همگنی واریانس را تایید کنیم زیرا در نمودار مانده ها در برابر پیش بینی پراکندگی حول میانگین ثابت است و در نمودار باکس پلات پراکندگی حول میانه ثابت است.

در ادامه به بررسی هم‌خطی بین متغیرهای مستقل پرداخته می‌شود.

```
correlation_matrix = X_train.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



در گام اول ماتریس همبستگی رسم می‌شود و مشاهده می‌شود که بین متغیرهای V , AT همبستگی مثبت بالایی وجود دارد همچنین بین متغیرهای AP , V و RH , AT هم همبستگی منفی قابل توجهی وجود دارد.

[251]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

X = add_constant(X)
vif_data = pd.DataFrame()
vif_data['Feature'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif_data)
```

```
  Feature           VIF
0   const  43761.151866
1      AT      5.977602
2       V      3.943003
3      AP      1.452639
4      RH      1.705290
```

در گام بعد به بررسی این موضوع با بررسی عامل تورم واریانس (VIF) پرداخته میشود و فرض میشود که اگر مقدار این آماره برابر ۱ باشد بدین معنی است که هیچ هم‌خطی وجود ندارد، اگر vif بین ۱ و ۵ باشد یعنی هم خطی متوسطی وجود دارد، اگر بین ۵ تا ۱۰ یعنی هم خطی شدید است و اکر بالای ۱۰ باشد به معنی وجود هم خطی بسیار شدیدی میان متغیرها می‌باشد.

در اینجا مقدار آماره vif برای متغیرهای V, AP, RH عددی بین ۱ تا ۵ شده که یعنی هم‌خطی بین این متغیرهای مستقل متوسط است و مقدار عددی این آماره برای متغیر AT عددی بین ۵ تا ۱۰ شده که نشانگر همخطی شدیدی می‌باشد.

پروژه ۲

برازش رگرسیون مولفه‌های اصلی بر مجموعه داده‌های داری هم‌خطی معنی‌دار

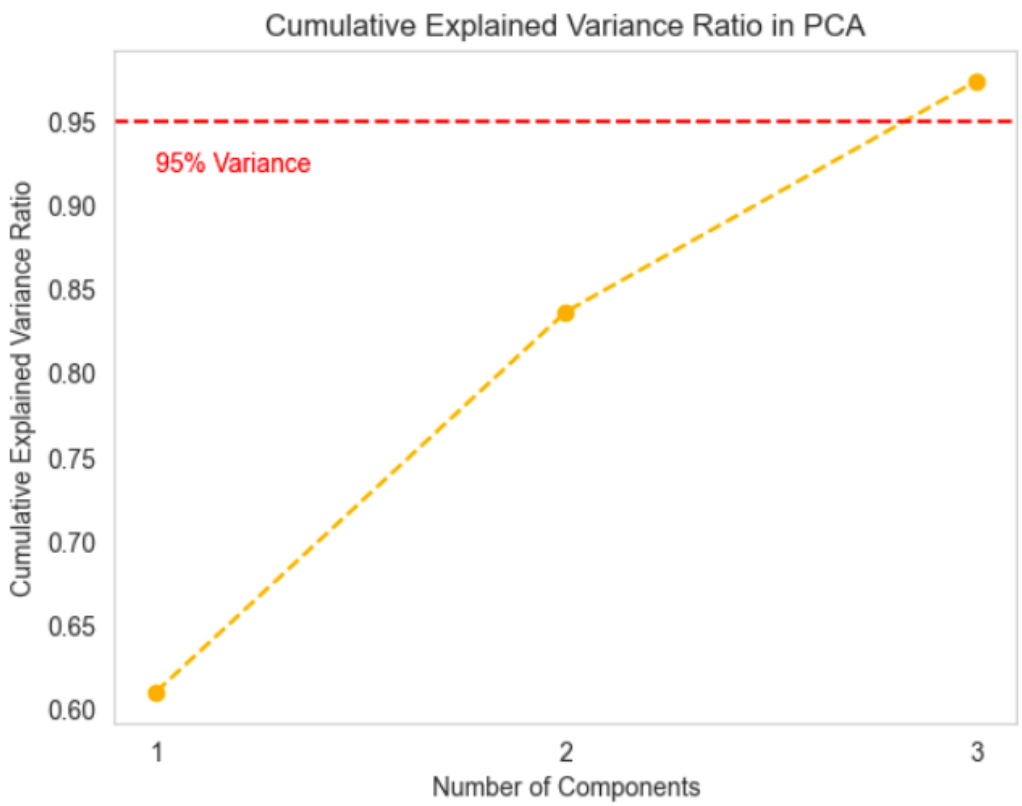
در ادامه پروژه ۱ برای همان مجموعه داده Combined Cycle Power Plant (نیروگاه سیکل ترکیبی) که ملاحظه شد هم‌خطی معناداری دارد، رگرسیون مولفه‌های اصلی برازش داده می‌شود.

```
[18]: from sklearn.decomposition import PCA

pca = PCA(n_components=0.95)
X_PCA = pca.fit_transform(X_train_scaled)
X_PCA = pca.transform(X_test_scaled)
print("Shape of Features Dataframe after PCA:", "\nTrain:", X_PCA.shape)
explained_variance = pca.explained_variance_ratio_
print("explained variance ratio", explained_variance)
cumulative_variance = np.cumsum(explained_variance)
print("cumulative variance ratio", cumulative_variance)
```

```
Shape of Train Featuers and Test Features Dataframe after PCA:
Train: (7654, 3)
Test: (1914, 3)
explained variance ratio [0.60997436 0.22680929 0.13751512]
cumulative variance ratio [0.60997436 0.83678365 0.97429877]
```

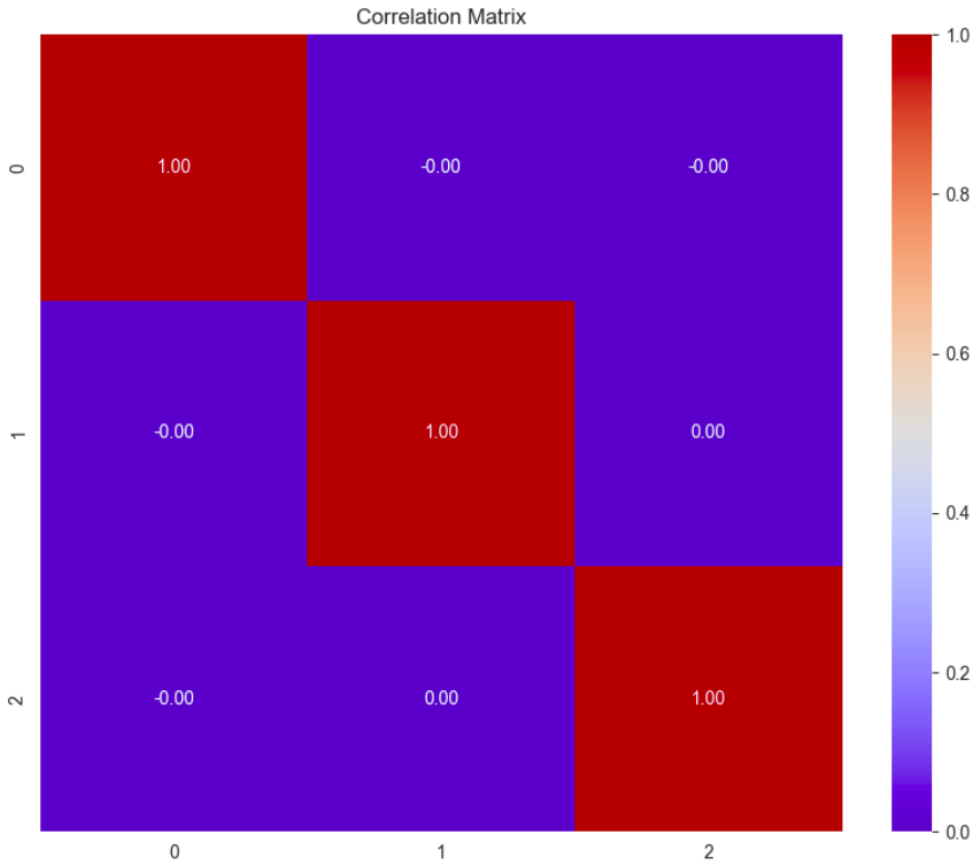
در این قسمت متد PCA با حفظ حداقل ۹۵٪ واریانس اجرا می‌شود که نرخ حفظ واریانس توسط اجزای اصلی(pc) با سه جزء اصلی توانسته حداقل ۹۵٪ واریانس را پوشش دهد. این موضوع را با جمع زدن اعدادی که در بخش explained variance ratio نمایش داده شده متوجه می‌شویم و اعداد پایین آن که مربوط به cumulative variance ratio است در واقع مقادیر تجمیعی همان اعداد بالا می‌باشد.که می‌بینیم بعد از جزء سوم این عدد به ۰.۹۷ رسیده است.

```
plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance, marker='o', linestyle='--', color='orange')
plt.axhline(y=0.95, color='r', linestyle='--')
plt.text(1, 0.92, '95% Variance', color='red')
plt.title('Cumulative Explained Variance Ratio in PCA')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.xticks(range(1, len(cumulative_variance) + 1))
plt.grid()
```



در این قسمت توضیحات بالا در یک نمودار برحسب مقادیر تجمیعی واریانس توضیح داده شده توسط مولفه‌های اصلی آورده می‌شود.

[258]:
```
PCA_Xdf = pd.DataFrame(X_PCA)
PCA_correlation_matrix = PCA_Xdf.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(PCA_correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



نمودار همبستگی پس از PCA به جهت نمایشی از رسیدن به هدف اعمال متد PCA رسم می‌شود که در آن ملاحظه می‌شود بین هیچ یک از مولفه‌های اصلی هیچ همبستگی وجود ندارد.

[259]:
```
PCA_model = LinearRegression()
PCA_model.fit(X_PCA, y)
PCA_y_pred = PCA_model.predict(X_PCA)
PCA_residuals = y - PCA_y_pred
print(f"Intercept: {PCA_model.intercept_:.3f}")
Co = [round(i, 3) for i in PCA_model.coef_]
print(f"Coefficients: {Co}")
PCA_R2 = r2_score(y, PCA_y_pred)
print(f"R^2 score: {PCA_R2:.3f}")
```

```
Intercept: 454.365
Coefficients: [-9.993, -1.539, 5.083]
R^2 score: 0.892
```

در اینجا پس از ایجاد مولفه‌های اصلی مدل رگرسیون خطی بر روی این داده ها پیاده می‌شود و ضرایب رگرسیونی روی داده‌ها و مقدار R2 بدست می‌آید و سپس مقدار بتا صفر و ضرایب رگرسیونی و R2 را نمایش داده می‌شود.

مشاهده می‌شود درصد ناچیزی از R2 مدل نسبته به مدل قبل از اعمال PCA کاهش یافته که به دلیل از دست رفتن قسمت ناچیزی از اطلاعات ناشی از اجرای این متد مورد انتظار است.

پروژه ۳

نگاشت ماتریسی ضرایب مدل رگرسیون مولفه‌های اصلی به ضرایب مناسب برای متغیرهای مستقل

در ادامه پروژه‌های ۱ و ۲ برای همان مجموعه داده Combined Cycle Power Plant (نیروگاه سیکل ترکیبی) که ملاحظه شد هم‌خطی معناداری دارد و رگرسیون مولفه‌های اصلی با روش‌های محاسبات ماتریسی برای آن برازش داده شد در این بخش به نمایشی از نگاشت ضرایب مدل پس از PCA متناظر با متغیرهای مستقل پیش از PCA و همچنین پیش از مقیاس بندی پرداخته می‌شود.

[61]:
```
PCA_coef = PCA_model.coef_
print("(PCA) Coefficients:\n", PCA_coef)
PCA_components = pca.components_
print("PCA Components:\n", PCA_components)
scaled_coefficients = np.dot(PCA_components.T, PCA_coef)
print("Coefficients for Scaled Features:\n", scaled_coefficients)
sigma = scaler.scale_
unscaled_coefficients = scaled_coefficients / sigma
print("Coefficients for Unscaled Features:\n", unscaled_coefficients)
```

```
(PCA) Coefficients:
 [-9.99049421 -1.53117512  5.09651488]
PCA Components:
 [[ 0.61439012  0.55946942 -0.40558288 -0.38081658]
 [-0.05185881  0.1051349  -0.64364381  0.75629357]
 [-0.16941701 -0.59206114 -0.63349575 -0.46844917]]
Coefficients for Scaled Features:
 [-6.92209235 -8.76780437  1.80888434  0.25906982]
Coefficients for Unscaled Features:
 [-0.93109243 -0.68989997  0.30426834  0.0177163 ]
```

در scaled coefficients با ضرب ترانهاده ماتریس مقادیر ویژه کوواریانس‌ها در ضرایب مدل رگرسیون پس از ایجاد مولفه های اصلی ضرایب مدل پس از PCA را به ضرایب داده های مقیاس بندی شده نگاشت دادیم و سپس در unscaled coefficients با تقسیم scaled coefficients بر انحراف استاندارد متغیرهای مستقل، ضرایب داده های مقباس بندی شده را به ضرایب مناسب برای متغیرهای مستقل نگاشت دادیم.

پروژه ۴

برازش رگرسیون های ریج (ستیغی) و لاسو و مولفه‌های اصلی و مقایسه MSE در این ۳ مدل برای داده‌های آزمون

در ادامه پروژه‌های قبل برای همان مجموعه داده Combined Cycle Power Plant (نیروگاه سیکل ترکیبی) که ملاحظه شد هم‌خطی معناداری دارد، این بخش علاوه بر تقسیم داده‌ها به دو قسمت داده‌های آموزش و داده‌های آزمون مدل‌های رگرسیون مولفه‌های اصلی، رگرسیون ریج (ستیغی) و رگرسیون لاسو برازش داده و مجموع مربعات خطای آنها مقایسه می‌شود.

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
data = pd.read_excel('Folds5x2_pp.xlsx')
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
print("Feautures\n", X.head())
print("Shape of Features Dataframe:", X.shape)
print("Targets\n", y.head())
print("Shape of Targets:", y.shape)
```

```
Feautures
       AT      V       AP     RH
0  14.96  41.76  1024.07  73.17
1  25.18  62.96  1020.04  59.08
2   5.11  39.40  1012.16  92.14
3  20.86  57.32  1010.24  76.64
4  10.82  37.50  1009.23  96.62
Shape of Features Dataframe: (9568, 4)
Targets
 0    463.26
1    444.37
2    488.56
3    446.48
4    473.90
Name: PE, dtype: float64
Shape of Targets: (9568,)
```

این دیتاست در ۹,۵۶۸ ردیف و ۵ ستون ارائه شده است.

از ستون اول تا چهارم داده‌هایمان را که شامل (AT),(V),(AP),(RH) است را در متغیر x می‌ریزیم که به عنوان متغیرهای مستقل ما هستند.

ستون پنجم داده‌هایمان را که مربوط به (PE) است را به عنوان متغیر پاسخ(وابسته) مدنظر قرار می‌دهیم و در متغیر y می‌ریزیم.

سپس ۵ سطر اول متغیرهای x, y را نمایش می‌دهیم.

در بخش بعد داده ها به دو قسمت آموزش و آزمون تقسیم‌بندی میشوند.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Train Feautures\n", X_train.head())
print("Shape of Train Features Dataframe:", X_train.shape)
print("Test Feautures\n", X_test.head())
print("Shape of Test Features Dataframe:", X_test.shape)
print("Train Targets\n", y_train.head())
print("Shape of Train Targets:", y.shape)
print("Test Targets\n", y_test.head())
print("Shape of Test Targets:", y_test.shape)
```

```
Train Feautures
          AT      V       AP     RH
5487  21.92  49.02  1009.29  88.56
3522  11.09  40.43  1025.47  74.97
6916   8.49  39.61  1021.05  87.74
7544  11.43  44.78  1013.43  82.45
7600  17.28  39.99  1007.09  74.25
Shape of Train Features Dataframe: (7654, 4)
Test Feautures
          AT      V       AP     RH
2513  19.64  48.06  1014.81  74.96
9411  28.26  69.23  1013.01  42.10
8745  27.98  67.17  1007.32  75.29
9085  28.64  69.23  1013.11  37.13
4950   9.34  38.08  1019.56  67.74
Shape of Test Features Dataframe: (1914, 4)
```

```
Train Targets
 5487    443.31
3522    490.96
6916    483.94
7544    471.09
7600    463.28
Name: PE, dtype: float64
Shape of Train Targets: (9568,)
Test Targets
 2513    455.27
9411    436.31
8745    440.68
9085    434.40
4950    482.06
Name: PE, dtype: float64
Shape of Test Targets: (1914,)
```

در این قسمت ما داده‌ها را با کمک تابع train_test_split به دو دسته Train و Test تقسیم کرده‌ایم که ۲۰ درصد داده‌ها را به Test و ۸۰ درصدشان را به Train اختصاص داده‌ایم. که نتیجه را به همراه ۵ ردیف اول از داده‌های train و test را مشاهده می‌کنیم.

در قسمت بعد مقیاس‌بندی استاندارد و پیاده سازی مدل رگرسیون خطی بدون هیچ تغییراتی بر داده‌های آموزش انجام می‌شود.

[5]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[6]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import  mean_squared_error, r2_score

model = LinearRegression()
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)
print(f"Intercept: {model.intercept_:.3f}")
Co = [round(i, 3) for i in model.coef_]
print(f"Coefficients: {Co}")
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse:.8f} (MSE Without Manipulating)")
r2 = r2_score(y_test, y_pred)
print(f"R^2 score: {r2:.8f}")
```

```
Intercept: 454.431
Coefficients: [-14.764, -2.95, 0.37, -2.312]
Mean Squared Error: 20.27370600 (MSE Without Manipulating)
R^2 score: 0.93010464
```

برای استاندارد سازی با استفاده از تابع Standard Scaler برروی داده‌های Train این کار را انجام می‌دهیم و در داده‌های Test مان از میانگین و واریانسی که از داده های Train استاندارد شده بدست آمده استفاده می‌کنیم.

و سپس یک مدل رگرسیون خطی پیاده کرده و به کمک تابع model.fit ضرایب رگرسیونی روی داده‌های آموزش بدست می‌آید و مقدار MSE و R2 بر داده‌های آزمون را محاسبه می‌کنیم و سپس مقدار بتا صفر و ضرایب رگرسیونی را نمایش می‌دهیم.

در قسمت بعد روش مولفه های اصلی بر داده‌های آموزش اعمال می‌شود و نتایج حاصل از آن به داده‌های آزمون تعمیم داده می‌شود.

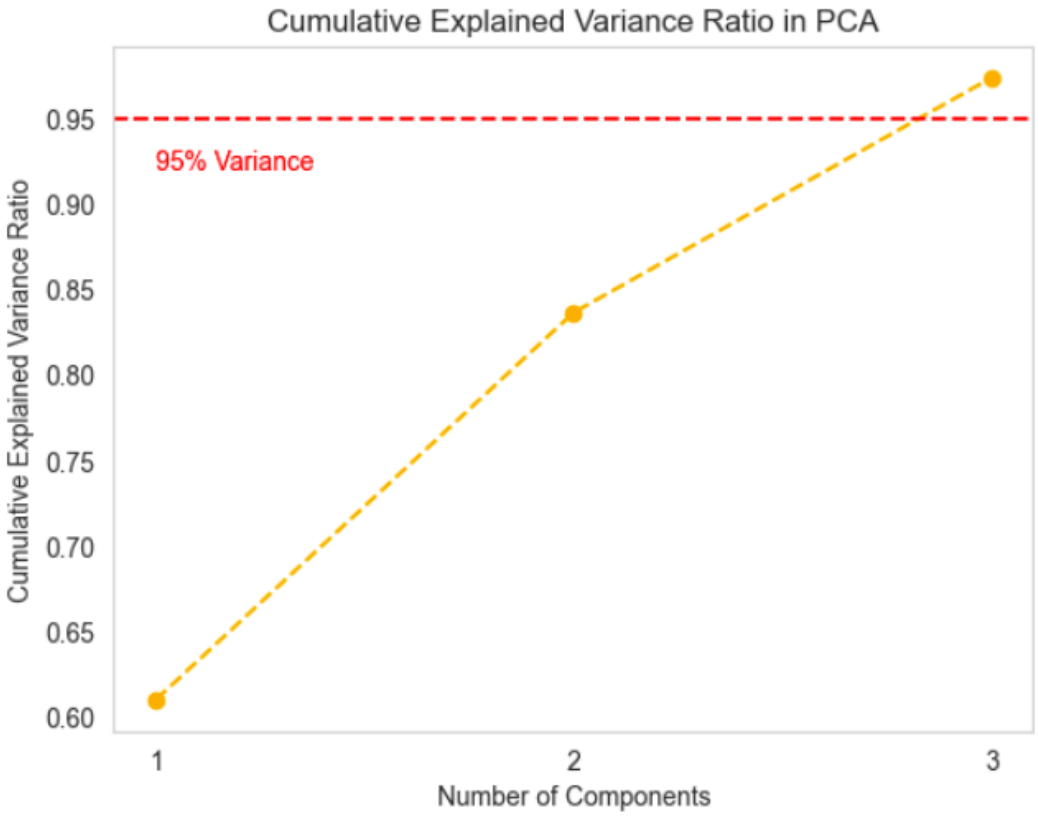
```
from sklearn.decomposition import PCA

pca = PCA(n_components=0.95)
X_train_PCA = pca.fit_transform(X_train_scaled)
X_test_PCA = pca.transform(X_test_scaled)
print("Shape of Train Featuers and Test Features Dataframe after PCA:", "\nTrain:", X_train_PCA.shape,"\nTest:", X_test_PCA.shape)
explained_variance = pca.explained_variance_ratio_
print("explained variance ratio", explained_variance)
cumulative_variance = np.cumsum(explained_variance)
print("cumulative variance ratio", cumulative_variance)
```

```
Shape of Train Featuers and Test Features Dataframe after PCA:
Train: (7654, 3)
Test: (1914, 3)
explained variance ratio [0.60997436 0.22680929 0.13751512]
cumulative variance ratio [0.60997436 0.83678365 0.97429877]
```

نرخ حفظ واریانس توسط اجزای اصلی(pc) برای حفظ حداقل ۹۵٪ تغییرات تنظیم شده است و این موضوع با سه جزء اصلی پوشش داده شده است.

این مطلب را با جمع زدن اعدادی که در بخش explained variance ratio نمایش داده شده متوجه می‌شویم و اعداد پایین آن که مربوط به cumulative variance ratio است در واقع مقادیر تجمیعی همان اعداد بالا می‌باشد. که می‌بینیم بعد از جزء سوم این عدد به ۰.۹۷ رسیده است.

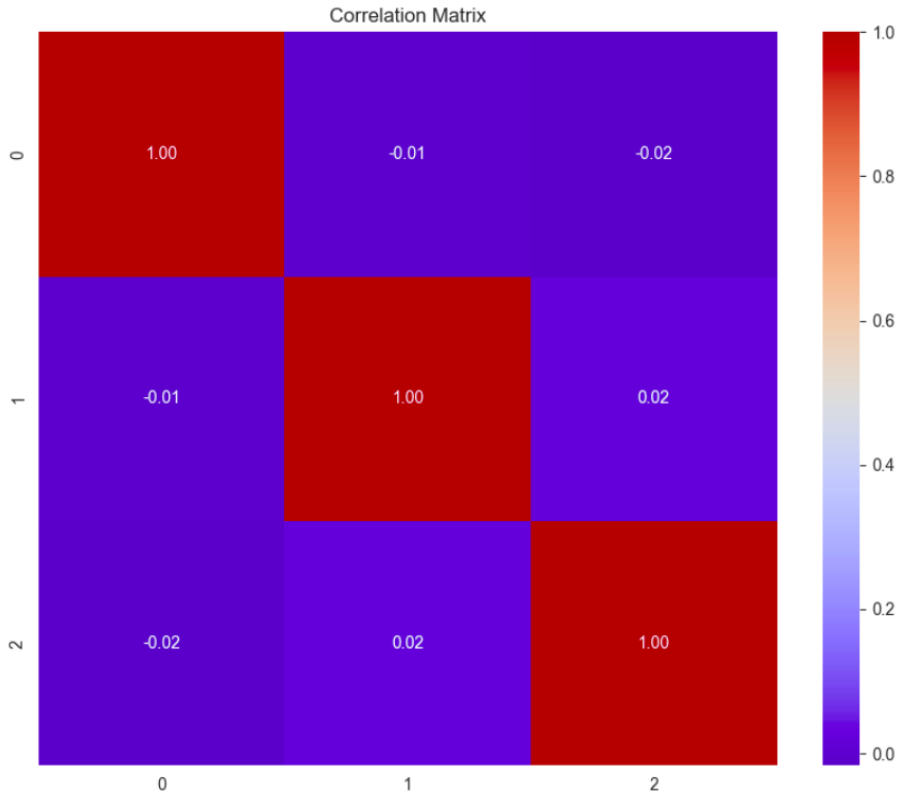
```
plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance, marker='o', linestyle='--', color='orange')
plt.axhline(y=0.95, color='r', linestyle='--')
plt.text(1, 0.92, '95% Variance', color='red')
plt.title('Cumulative Explained Variance Ratio in PCA')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.xticks(range(1, len(cumulative_variance) + 1))
plt.grid()
```



در اینجا توضیحات بالا در یک نمودار برحسب مقادیر تجمیعی واریانس توضیح داده شده توسط مولفه‌های اصلی آورده می‌شود.

در بخش بعد میزان موثر بودن روش PCA بر داده‌های تست به کمک ماتریس همبستگی این قسمت از داده‌ها پس از PCA بررسی می‌شود.

```
PCA_Xdf = pd.DataFrame(X_test_PCA)
PCA_correlation_matrix = PCA_Xdf.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(PCA_correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



ملاحظه می‌شود که در این بخش از داده‌ها با اعمال ضرایب حاصل از اجرای PCA در بخش داده‌های آموزش، ناهمبستگی بین متغیرهای مستقل حاصل هنوز برقرار بوده و همبستگی‌ها مقادیر بسیار ناچیزی هستند.

در قسمت بعد مدل رگرسیون مولفه‌های اصلی بر داده‌های آموزش برازش داده می‌شود و نتایج آن برای داده‌های تست بررسی می‌شود.

```python
PCA_model = LinearRegression()
PCA_model.fit(X_train_PCA, y_train)
PCA_y_pred = PCA_model.predict(X_test_PCA)
PCA_residuals = y_test - PCA_y_pred
print(f"Intercept: {PCA_model.intercept_:.3f}")
Co = [round(i, 3) for i in PCA_model.coef_]
print(f"Coefficients: {Co}")
mse = mean_squared_error(y_test, PCA_y_pred)
print(f"Mean Squared Error: {mse:.8f} (MSE After PCA)")
PCA_R2 = r2_score(y_test, PCA_y_pred)
print(f"R^2 score: {PCA_R2:.8f}")
```

```
Intercept: 454.431
Coefficients: [-9.99, -1.531, 5.097]
Mean Squared Error: 30.33164515 (MSE After PCA)
R^2 score: 0.89542903
```

پس از برازش شاهد افزایش MSE و کاهش R2 در پیش‌بینی داده‌های آزمون نسبت به رگرسیون داده‌های اولیه هستیم که این تغییرات طبیعتا ناشی از، از دست دادن مقدار جزئی از اطلاعات است.

در بخش بعد ابتدا توضیحاتی درباره رگرسیون‌های ریج و لاسو داده می‌شود و سپس این مدل‌ها بر داده‌های پروژه برازش داده می‌شوند.

با فرض استفاده از RSS(Residual Sum of Squares) یا جمع توان دوم مانده‌ها، مجموع مربع مانده ها را حساب می‌کنیم.

می‌خواهیم یک penalty term برای این تابع(RSS) بر حسب پارامترهای مدل تعریف کنیم. که در اینجا دو روش ارائه میشود و هر کدام منتج به یکی از رگرسیون های ریج و لاسو می‌شوند.

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

روش اول : رگرسیون ریج

در رگرسیون ریج در واقع می‌خواهیم یک عامل محدود کننده برای بزرگ شدن پارامترهای مدل تعیین کنیم.

که از مجموع توان دوم ضرایب با ضریب لاندا به عنوان penalty term استفاده می‌کنیم.

$$\beta_0{}^2 + \beta_1{}^2 + \cdots + \beta_p{}^2 \leq C^2$$

نحوه محاسبه ضرایب جدید پس از اضافه کردن penalty term بدین شرح خواهد بود:

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

$$RSS_{Ridge} = (\mathbf{Y} - \mathbf{X}\beta)^T (\mathbf{Y} - \mathbf{X}\beta) + \lambda \beta^T \beta = ||\mathbf{Y} - \mathbf{X}\beta||^2 + \lambda ||\beta||^2$$

یک تعبیر جبرخطی از اضافه کردن این عبارت محدود کننده(penalty term) آن است که در صورت وارون ناپذیر بودن $\mathbf{X}^T\mathbf{X}$ با اضافه کردن مقادیر $\lambda\mathbf{I}$ به آن ماتریس حاصل وارون پذیر خواهد شد و پیدا کردن کمینه مجموع مربعات امکان پذیر خواهد شد.

$$\nabla RSS_{\text{Ridge}} = -2\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\beta) + 2\lambda\mathbf{I}\beta = 0$$

$$\begin{aligned} -2\left(\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\beta) - \lambda\mathbf{I}\beta\right) &= 0 \\ \mathbf{X}^T(\mathbf{Y} - \mathbf{X}\beta) - \lambda\mathbf{I}\beta &= 0 \\ \mathbf{X}^T\mathbf{Y} - \mathbf{X}^T\mathbf{X}\beta + \lambda\mathbf{I}\beta &= 0 \\ (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\beta &= \mathbf{X}^T\mathbf{Y} \end{aligned} \qquad \beta = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$$

روش دوم : رگرسیون لاسو

در رگرسیون لاسو در واقع می‌خواهیم یک عملگر گزینش و انقباض کمترین قدرمطلق تعیین کنیم. که با کاهش ضرایب خاصی از مدل به صفر، تأثیر آنها در پیش‌بینی نهایی را از میان می برد.

که از مجموع قدرمطلق ضرایب با ضریب لاندا به عنوان penalty term استفاده می‌کنیم.

$$\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j} |\beta_j|$$

نحوه محاسبه ضرایب جدید پس از اضافه کردن penalty term بدین شرح است که به دلیل وجود پنالتی L1، مشتق تابع هزینه با سه حالت مواجه می‌شود:

$$0 = \text{sign}(\beta_j) \cdot \lambda + {}_{ij}x\,(\hat{}_i y - {}_i y) \sum_{i=1}^{n} \frac{1}{n} - = \frac{J(\beta)\partial}{{}_j\beta\partial}$$

1. برای $0 < {}_j\beta$: مشتق به صورت زیر است:

$$0 = \lambda + {}_{ij}x\,(\hat{}_i y - {}_i y) \sum_{i=1}^{n} \frac{1}{n} - = \frac{J(\beta)\partial}{{}_j\beta\partial}$$

2. برای $0 > {}_j\beta$: مشتق به صورت زیر است:

$$0 = \lambda - {}_{ij}x\,(\hat{}_i y - {}_i y) \sum_{i=1}^{n} \frac{1}{n} - = \frac{J(\beta)\partial}{{}_j\beta\partial}$$

3. برای $0 = {}_j\beta$: اگر مقدار جریمه‌ی λ بزرگ‌تر از مقدار مطلق گرادیان باشد، ${}_j\beta$ برابر صفر خواهد شد (چون این معادله برای صفر بهینه می‌شود).

در بخش بعد به پیاده‌سازی برازش این مدل‌ها پرداخته می‌شود، شایان ذکر است که برای یافتن بهترین مقدار پارامتر جریمه از تکنیک cross validation استفاده می‌شود.

```
from sklearn.linear_model import Ridge, Lasso
from sklearn.model_selection import GridSearchCV
alpha_range = np.logspace(-4, 4, 50)
print("Alpha Range:\n", alpha_range)
```

```
[1.00000000e-04 1.45634848e-04 2.12095089e-04 3.08884360e-04
 4.49843267e-04 6.55128557e-04 9.54095476e-04 1.38949549e-03
 2.02358965e-03 2.94705170e-03 4.29193426e-03 6.25055193e-03
 9.10298178e-03 1.32571137e-02 1.93069773e-02 2.81176870e-02
 4.09491506e-02 5.96362332e-02 8.68511374e-02 1.26485522e-01
 1.84206997e-01 2.68269580e-01 3.90693994e-01 5.68986603e-01
 8.28642773e-01 1.20679264e+00 1.75751062e+00 2.55954792e+00
 3.72759372e+00 5.42867544e+00 7.90604321e+00 1.15139540e+01
 1.67683294e+01 2.44205309e+01 3.55648031e+01 5.17947468e+01
 7.54312006e+01 1.09854114e+02 1.59985872e+02 2.32995181e+02
 3.39322177e+02 4.94171336e+02 7.19685673e+02 1.04811313e+03
 1.52641797e+03 2.22299648e+03 3.23745754e+03 4.71486636e+03
 6.86648845e+03 1.00000000e+04]
```

```
ridge = Ridge()
ridge_params = {'alpha': alpha_range}
ridge_cv = GridSearchCV(ridge, ridge_params, cv=10)
ridge_cv.fit(X_train, y_train)
best_alpha_ridge = ridge_cv.best_params_['alpha']
print(f"Best alpha for Ridge: {best_alpha_ridge:.5f}")
```

```
Best alpha for Ridge: 35.56480
```

```
lasso = Lasso()
lasso_params = {'alpha': alpha_range}
lasso_cv = GridSearchCV(lasso, lasso_params, cv=10)
lasso_cv.fit(X_train, y_train)
best_alpha_lasso = lasso_cv.best_params_['alpha']
print(f"Best alpha for Lasso: {best_alpha_lasso:.8f}")
```

```
Best alpha for Lasso: 0.00294705
```

با استفاده از تابع logspace یک بازه ۵۰ تایی بصورت تصاعد هندسی از ۰.۰۰۱ تا ۱۰۰۰ برای یافتن بهترین مقدار در این تکنیک ساخته می‌شود و سپس برای هر یک از انواع رگرسیون‌های ریج و لاسو با استفاده از تکنیک cross validation بهترین مقدار پارامتر جریمه پیدا می‌شود.

```
final_ridge_model = Ridge(alpha=best_alpha_ridge)
final_ridge_model.fit(X_train, y_train)
y_pred_ridge = final_ridge_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred_ridge)
print(f"Mean Squared Error: {mse:.8f} (MSE after Ridge Regularization)")
r2_ridge = r2_score(y_test, y_pred_ridge)
print(f"R² score for Ridge: {r2_ridge:.8f}")
```

```
Mean Squared Error: 20.27295055 (MSE after Ridge Regularization)
R² score for Ridge: 0.93010725
```

```
lasso_final = Lasso(alpha=best_alpha_lasso)
lasso_final.fit(X_train, y_train)
y_pred_lasso = lasso_final.predict(X_test)
mse = mean_squared_error(y_test, y_pred_lasso)
print(f"Mean Squared Error: {mse:.8f} (MSE after Lasso Regularization)")
r2_lasso = r2_score(y_test, y_pred_lasso)
print(f"R² score for Lasso: {r2_lasso:.8f}")
```

```
Mean Squared Error: 20.27393151 (MSE after Lasso Regularization)
R² score for Lasso: 0.93010387
```

مدل‌های رگرسیون ریج و لاسو را با پارامتر تعیین شده در تکنیک بر داده های آموزش پیاده می‌کنیم بعد برروی داده‌های آزمون آنرا بررسی می‌کنیم. ملاحضه می‌کنیم که MSE در این دو مدل تغییر بسیار ناچیزی نسبت به مدل برازش داده شده به داده‌های دستکاری نشده اولیه داشته که قابل ملاحظه نیست و نشان‌دهنده این موضوع است که داده‌های پروژه نسبت به مدل رگرسیون خطی دچار بیش‌برازش (overfitting) نمی‌شوند، گرچه مقدار آنها نسبت به مقدار MSE مدل رگرسیون مولفه‌های اصلی کاهش داشته که به موجب از دست رفتن جزئی اطلاعات در مدل رگرسیون مولفه‌های اصلی پیش آمده است.

پروژه ۵

تحلیل عاملی

بخشی از مجموعه داده‌های Diabetes Dataset (originally from the N. Inst. of Diabetes & Diges. & Kidney Dis) از پایگاه kaggle برای پروژه اختیار گردیده است.

این مجموعه داده که متشکل از ۶۹۱۲ نمونه است از پایگاه داده‌های موسسه ملی دیابت و بیماری های گوارشی و کلیوی ایالات متحده تهیه شده است.

در انتخاب این مجموعه داده چندین محدودیت برای انتخاب این نمونه ها از یک پایگاه داده بزرگتر قرار داده شده است. به طور خاص، همه موارد مورد بررسی در اینجا زنان حداقل ۲۱ ساله هستند که به گروه بومی Pima Indians تعلق دارند.

متغیرهای مجموعه داده‌ها:

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- Blood Pressure: Diastolic blood pressure (mm Hg)
- Skin Thickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- Diabetes Pedigree Function: Diabetes pedigree function
- Age: Age (years)
- Outcome: Class variable (0 or 1)

(دفعات بارداری، سطح گلوکز، فشار خون، ضخامت پوست، سطح انسولین، شاخص توده بدنی، سابقه خانوادگی دیابت، سن، طبقه نمونه شامل سالم یا بیمار)

با توجه به عدم وجود اطلاعات پیشین در خصوص همبستگی متغیرها نوع تحلیل عاملی در این پروژه از نوع اکتشافی (EFA) است.

در گام اول پکیج‌های مورد نیاز و داده‌ها فراخوانی میشوند.

[183]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

...

[184]:
```
df = pd.read_csv("diabetes.csv")
df.shape
```

[184]:
```
(768, 9)
```

[207]:
```
data = df.drop('Outcome',axis=1)
```

متغیر خروجی (outcome) از مجموعه داده‌ها حذف می‌شود.

در بخش بعد یک مرحله EDA روی داده‌های پروژه بررسی می‌شود.

```
data = df.drop('Outcome',axis=1)
```

```
data.info(verbose=False)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Columns: 8 entries, Pregnancies to Age
dtypes: float64(2), int64(6)
memory usage: 48.1 KB
```

```
data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

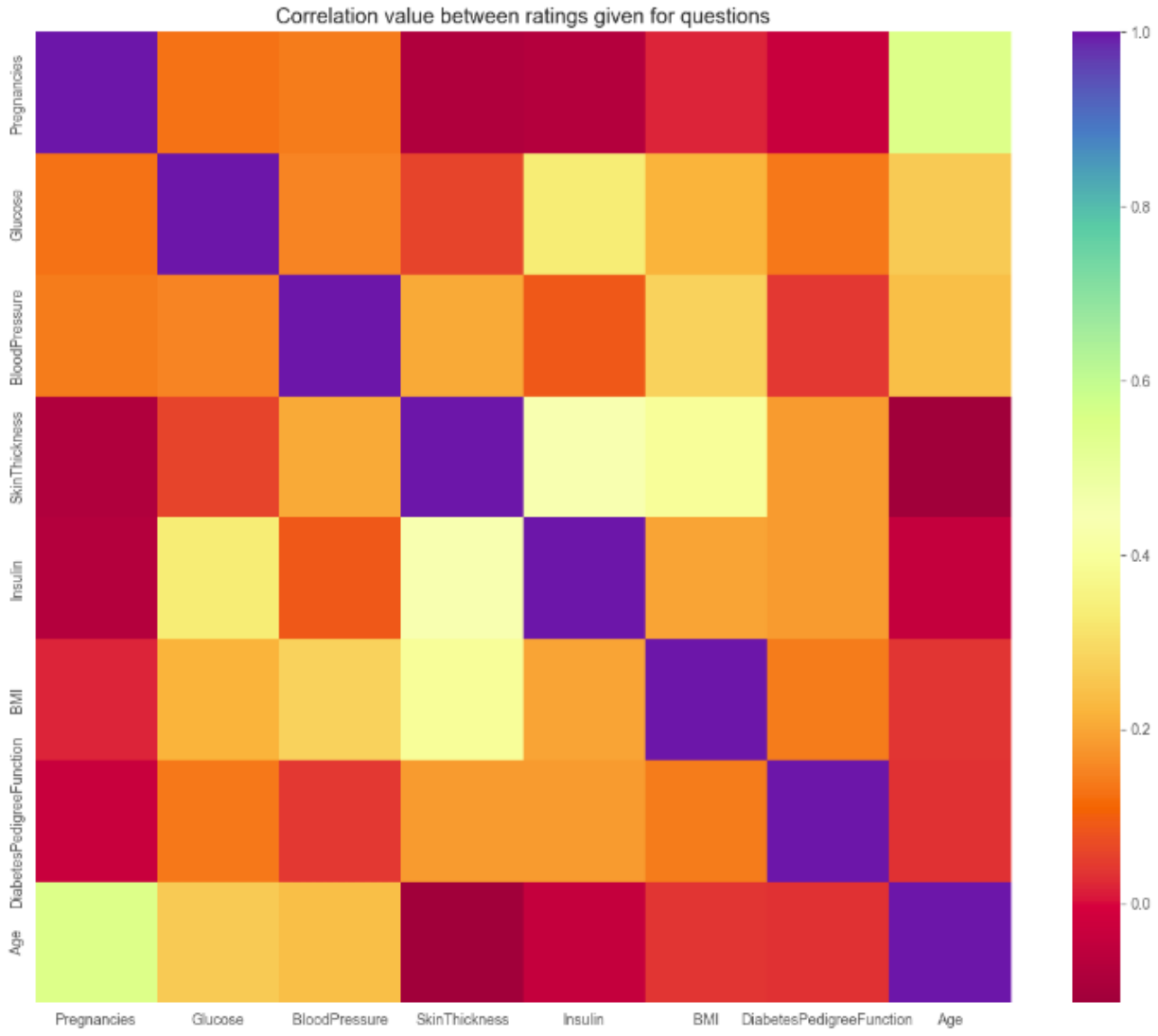
```
missing_values = data.isnull().sum()
print(missing_values[missing_values > 0])
```

```
Series([], dtype: int64)
```

اطلاعات دیتافریم را با استفاده از data.info مشاهده می‌شود. ۵ سطر اول دیتافریم را می‌بینیم و وجود داده های از دست رفته را بررسی می‌کنیم که مشاهده می‌شود داده از دست رفته ای وجود ندارد.

```
plt.figure(figsize=(15,12))
sns.heatmap(data.corr(),cmap='Spectral')
plt.title("Correlation value between ratings given for questions")
```

```
Text(0.5, 1.0, 'Correlation value between ratings given for questions')
```



در این مرحله نمودار heatmap را برای ملاحظه همبستگی دو به دوی متغیرهای مجموعه داده ها رسم می کنیم. ملاحظه می شود که همبستگی های قابل توجه و بعضاً شدیدی در بین متغیرهای مجموعه داده وجود دارد.

در بخش بعد ابتدا توضیحاتی درباره آزمون‌های بارتلت و KMO در جهت تعیین کفایت داده‌ها برای پیاده‌سازی تحلیل عاملی داده می‌شود و سپس این آزمون‌ها برای داده‌های پروژه بررسی می‌شوند.

در بررسی و تحقیقاتی که برمبنای تحلیل عاملی اکتشافی(EFA) انجام می‌شود دو آزمون بارتلت و KMO در تحلیل عاملی به محققین این امکان را می‌دهد که قبل از اجرا از بسندگی یا کفایت حجم نمونه و وجود همبستگی مناسب بین متغیرها مطمئن شده، سپس تحلیل عاملی را به کار برند.

آزمون بارتلت:

فرض اینکه واریانس‌ها در گروه‌ها یا نمونه‌ها برابر هستند، با آزمون بارتلت قابل رد است. در تحلیل عاملی، ما به یک ماتریس همبستگی نیاز داریم که به اندازه کافی همبستگی بین متغیرها را نشان دهد تا بتوان عوامل پنهان را استخراج کرد. اگر همبستگی‌ها خیلی پایین باشند، تحلیل عاملی چندان مفید نخواهد بود.

به این ترتیب از آزمون بارتلت بر اساس آماره‌ای است که توزیع نمونه‌ای آن تقریباً یک توزیع کای ۲ (Chi-Square) با (k-1) درجه آزادی است بهره می‌بریم. بطوری که k نمایانگر تعداد نمونه‌های تصادفی است و ممکن است در هر نمونه اندازه‌ای متفاوت داشته باشد. از طرفی توزیع هر یک از جوامع نیز نرمال فرض شده و استقلال نیز در این حالت منظور گردیده است.

اگر از k جامعه نمونه‌هایی به اندازه ni داشته باشیم و $Si2$ نشانگر واریانس جامعه i ام باشد، آنگاه آماره بارتلت به صورت زیر است.

$$\chi^2 = \frac{(N-k)\ln(S_p^2) - \sum_{i=1}^{k}(n_i-1)\ln(S_i^2)}{1+\frac{1}{3(k-1)}\left(\sum_{i=1}^{k}(\frac{1}{n_i-1}) - \frac{1}{N-k}\right)}$$

که در آن:

$$S_p^2 = \frac{1}{N-k}\sum_i (n_i-1)S_i^2 \qquad N = \sum_{i=1}^{k} n_i$$

آماره آزمون بارتلت که به صورت نسبت مربع دو توزیع نرمال استاندارد مشخص شده، به طور مجانبی، توزیع کای ۲ با k-1 درجه آزادی تحت فرض صفر خواهد بود. به این ترتیب اگر مقدار آماره از صدک α ام چنین توزیعی بزرگتر باشد، فرض صفر را می‌کنیم.

$$\chi^2 > \chi^2_{k-1,\alpha}$$

در این بخش این آزمون برای داده‌های پروژه بررسی می‌شود.

```
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
chi2,p = calculate_bartlett_sphericity(data)
print("Chi squared value : ",chi2)
print("p value : ",p)
```

```
Chi squared value :  948.2262232122048
p value :  1.2575496243955591e-181
```

به کمک تابع calculate_bartlett_sphericity آزمون بارتلت را اجرا کرده و فرض همگنی واریانس قاطعانه رد می شود و نتیجه می گیریم برای تحلیل عاملی مناسب است.

آزمون KMO:

آزمون KMO (Kaiser-Meyer-Olkin) یا شاخص کفایت نمونه‌برداری در تحلیل عاملی استفاده می‌شود تا مناسب بودن داده‌ها برای تحلیل عاملی را از نظر کفایت نمونه‌برداری بررسی کند. این آزمون بررسی می‌کند که آیا حجم نمونه و میزان همبستگی بین متغیرها برای کشف ساختارهای پنهان یا عوامل کافی است یا خیر. مقدار شاخص KMO بین ۰ و ۱ متغیر است و برای تعیین کفایت داده‌ها معیارهای زیر معمولاً استفاده می‌شوند:

KMO بالای ۰.۹۰ :عالی

KMO بین ۰.۸۰ و ۰.۸۹ :خوب

KMO بین ۰.۷۰ و ۰.۷۹ :متوسط

KMO بین ۰.۶۰ و ۰.۶۹ :قابل قبول

KMO بین ۰.۵۰ و ۰.۵۹ :ضعیف، ولی می‌تواند قابل قبول باشد

KMO کمتر از ۰.۵۰ :نامناسب، داده‌ها برای تحلیل عاملی کفایت نمی‌کنند

در این بخش این آزمون برای داده‌های پروژه بررسی می‌شود.

```
from factor_analyzer.factor_analyzer import calculate_kmo
kmo_vars,kmo_model = calculate_kmo(data)
print(kmo_model)
```

```
0.5889870819164513
```

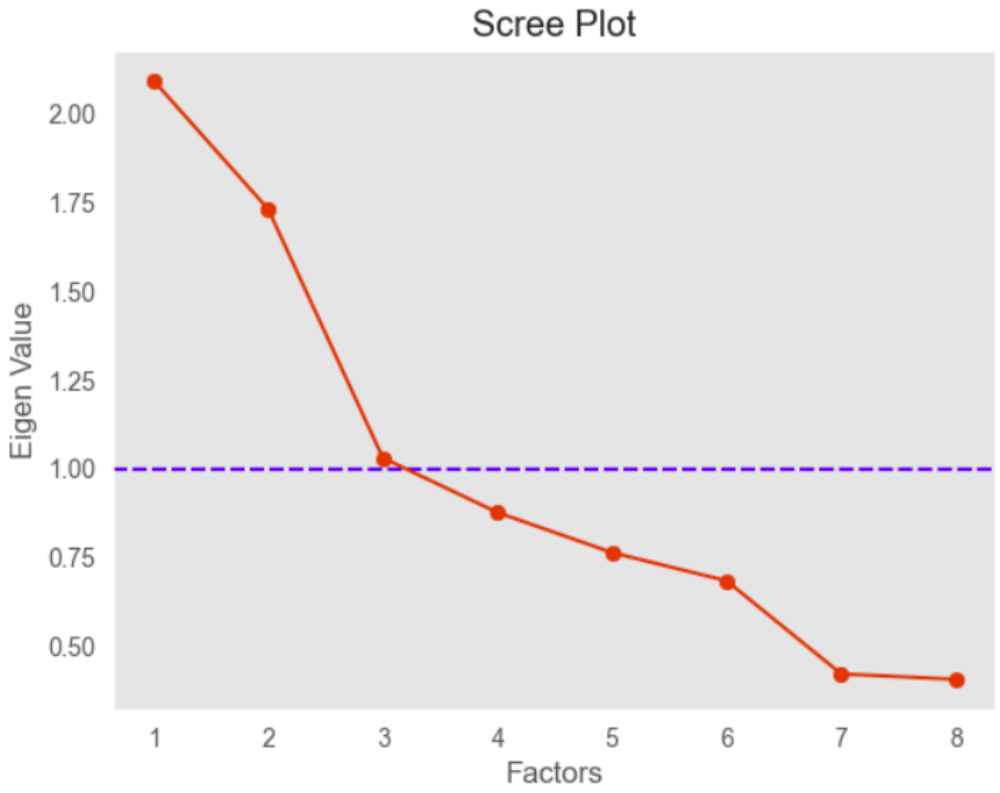
با کمک تابع calculate_kmo این آزمون نیز اجرا شد و با توجه به مقدار آماره این آزمون نتیجه قابل قبولی برای پیاده سازی تحلیل عاملی می گیریم.

در بخش‌های بعد به پیاده سازی تحلیل عاملی پرداخته می‌شود. شایان ذکر است این موضوع با استفاده از تابع factorAnalyzer صورت می‌گیردکه روش پیش‌فرض آن روش پیاده سازی مبتنی بر روش حداکثر درست نمایی است به دنبال ماکسیمم کردن تابع درستنمایی زیر است.

$$l(\mu, \mathbf{L}, \mathbf{\Psi}) = -\frac{np}{2}\log 2\pi - \frac{n}{2}\log|\mathbf{LL}' + \mathbf{\Psi}| - \frac{1}{2}\sum_{i=1}^{n}(\mathbf{X_i} - \mu)'(\mathbf{LL}' + \mathbf{\Psi})^{-1}(\mathbf{X_i} - \mu)$$

در قسمت بعد یافتن تعداد عوامل بررسی می‌شود.

```
from factor_analyzer import FactorAnalyzer
n = data.shape[1]
fa = FactorAnalyzer(rotation = None,n_factors=n)
fa.fit(data)
ev,_ = fa.get_eigenvalues()
plt.scatter(range(1,n+1),ev)
plt.axhline(y=1, color='b', linestyle='--')
plt.plot(range(1,n+1),ev)
plt.title('Scree Plot')
plt.xlabel('Factors')
plt.ylabel('Eigen Value')
plt.grid()
```



یکبار تحلیل عاملی را بدون دوران و کاهش بعد برای مجموعه داده ها اجرا کرده و مقادیر ویژه حاصل از ماتریس بارهای عاملی را محاسبه می کنیم تا به اهمیت عامل پی ببریم. ملاحظه می شود که ۳ عامل مقدار ویژه بالای ۱ دارند که در scree plot رسم شده قابل مشاهده است. بنابراین برای تحلیل عاملی ۳ عامل پنهان را در نظر می گیریم.

- قاعده Kaiser پیشنهاد می‌کند که عوامل با مقدار ویژه کمتر از ۱ نباید به عنوان عوامل جداگانه در نظر گرفته شوند اگر مقدار ویژه یک عامل کمتر از ۱ باشد، این عامل حتی به اندازه یک متغیر منفرد واریانس داده‌ها را توضیح نمی‌دهد و به اصطلاح نمی‌تواند به‌طور قابل قبولی «توضیح‌دهنده» داده‌ها باشد.

در بخش بعد تحلیل عاملی نهایی پیاده‌سازی می‌شود.

```
fa = FactorAnalyzer(n_factors=3,rotation='varimax')
fa.fit(df)
fa_load = pd.DataFrame(fa.loadings_,index=df.columns)
```

```
loadings = pd.DataFrame(fa.loadings_)
loadings.rename(columns = lambda x: 'Factor-' + str(x + 1), inplace=True)
loadings.index = df.columns
loadings
```

	Factor-1	Factor-2	Factor-3
Pregnancies	-0.052151	0.659111	0.069804
Glucose	0.125628	0.144566	0.808942
BloodPressure	0.296692	0.280431	0.056146
SkinThickness	0.887929	-0.084547	-0.041905
Insulin	0.445642	-0.121957	0.300707
BMI	0.475820	0.085185	0.219497
DiabetesPedigreeFunction	0.218356	-0.020511	0.191929
Age	-0.047681	0.790515	0.175055

تحلیل عاملی با تعیین ۳ عامل پنهان و نوع دوران از نوع واریمکس پیاده سازی می‌شود و ماتریس بارهای عاملی آن مشخص می‌شوند.

```
segments = loadings[loadings >= .4].fillna(loadings[loadings <= -.4])
segments
```

	Factor-1	Factor-2	Factor-3
Pregnancies	NaN	0.659111	NaN
Glucose	NaN	NaN	0.808942
BloodPressure	NaN	NaN	NaN
SkinThickness	0.887929	NaN	NaN
Insulin	0.445642	NaN	NaN
BMI	0.475820	NaN	NaN
DiabetesPedigreeFunction	NaN	NaN	NaN
Age	NaN	0.790515	NaN

در این قسمت از بارهای عاملی کمتر از ۰/۴ صرف نظر می‌شود تا در یک نمای واضح تر مشخص شود هر یک از متغیر ها به کدام عامل تعلق دارند.

```
communalities = fa.get_communalities()
communalities = pd.DataFrame(communalities, index=data.T.index, columns=['communalities'])
communalities
```

	communalities
Pregnancies	0.324363
Glucose	0.203466
BloodPressure	0.214003
SkinThickness	0.399474
Insulin	0.523554
BMI	0.514451
DiabetesPedigreeFunction	0.066635
Age	0.900148

در این قسمت پس از استخراج عوامل، اشتراک‌ها (Communalities) نشان می‌دهد که چه مقدار از واریانس هر متغیر توسط عوامل مشترک توضیح داده شده است و اهمیت متغیرها در هر یک از عوامل پنهان چه میزان است.

```
# Check variance
factorVariance = pd.DataFrame(fa.get_factor_variance())
factorVariance.rename(columns = lambda x: 'Factor-' + str(x + 1), inplace=True)
factorVariance.index = ['SS Loadings', 'Proportion Variance', 'Cumulative Variance']
factorVariance
```

	Factor-1	Factor-2	Factor-3
SS Loadings	1.383003	1.241710	0.999559
Proportion Variance	0.172875	0.155214	0.124945
Cumulative Variance	0.172875	0.328089	0.453034

و در این بخش بارهای عاملی با توجه به مقدار واریانس توضیح داده شده توسط آنها بررسی می‌شوند و مقادیر تجمیعی آنها ملاحظه می‌شود.

در اینجا یک نکته مهم شایان ذکر است:

اگرچه تحلیل عاملی درصد نه چندان زیادی از واریانس داده‌ها را توضیح می‌دهد، اما کفایت آن به دلیل توانایی شناسایی عوامل مؤثر و معنادار تایید می‌شود. در ادامه از این عوامل در یک مدل رگرسیون لجستیک استفاده شده و نتایج قابل‌قبولی ارائه می‌شود، که نشان‌دهنده ارتباط قوی این عوامل با متغیر هدف (یعنی خروجی: بیمار یا سالم) است. بنابراین، تحلیل عاملی توانسته با وجود واریانس توضیح داده‌شده محدود، ساختاری مناسب برای پیش‌بینی و مدل‌سازی ایجاد کند.

در ادامه به نمایش پیاده‌سازی مطلب بالا پرداخته میشود.

[22]:
```
X = df.copy().drop('Outcome',axis=1)
y = df['Outcome'].copy()
X.shape, y.shape
```

[22]:
```
((768, 8), (768,))
```

[23]:
```
from sklearn.decomposition import FactorAnalysis
transformer = FactorAnalysis(n_components=3, rotation='varimax', random_state=0)
X_transformed = transformer.fit_transform(X)
X_transformed.shape
```

[23]:
```
(768, 3)
```

[24]:
```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

k = 10
model = LogisticRegression()
scores = cross_val_score(model,X_transformed,y,cv=k)
print(f"{k} fold - cross validated scores: {scores}")
print(f"Average accuracy scores: {scores.mean()}")
```

```
10 fold - cross validated scores: [0.72727273 0.67532468 0.71428571 0.68831169 0.75324675 0.75324675
 0.76623377 0.77922078 0.77631579 0.77631579]
Average accuracy scores: 0.7409774436090226
```

در این بخش پس از ایجاد متغیرهای مستقل و متغیر وابسته در دو متغیر جداگانه، تحلیل عاملی بر روی متغیرهای مستقل پیاده‌سازی می‌شود و از نتایج آن یعنی عامل‌های ایجاد شده به عنوان متغیر مستقل در یک برازش مدل رگرسیون لجستیک استفاده می‌شود، نتیجه دقت حاصل از این مدل رگرسیون لجستیک با استفاده از تکنیک cross validation برای ۱۰ بخش محاسبه و میانگین آن مشخص می‌شود که دقت قابل قبولی را باتوجه به واریانس توضیح داده شده توسط عوامل تحلیل عاملی ارائه می‌دهد.