به نام خدا

مستندات پروژه اول

مدل رگرسیون برای پیشبینی اعتبار مالی افراد

مبانی یادگیری ماشین

رویا صالحی

۴۰۰۳۶۲۳۰۲۳

دکتر محمد کیانی

بهار ۱۴۰۳

دیتاست:

در ابتدا تمامی کتابخانه های مورد نیاز **install** و **import** شدند.

پس از خواندن دیتاست، برای مطلع شدن از وضعیت دیتاها، از دستورات زیر استفاده شد:

df.info()

تعداد دیتای ویژگی‌هایی که مقادیر نال دارند، با دیتاتایپ آنان شناسایی شد.

df.describe()

اطلاعات کلی شامل میانگین، مقدار ماکسیمم، مینیمم، میزان پراکندگی داده ها و... شناسایی شد.

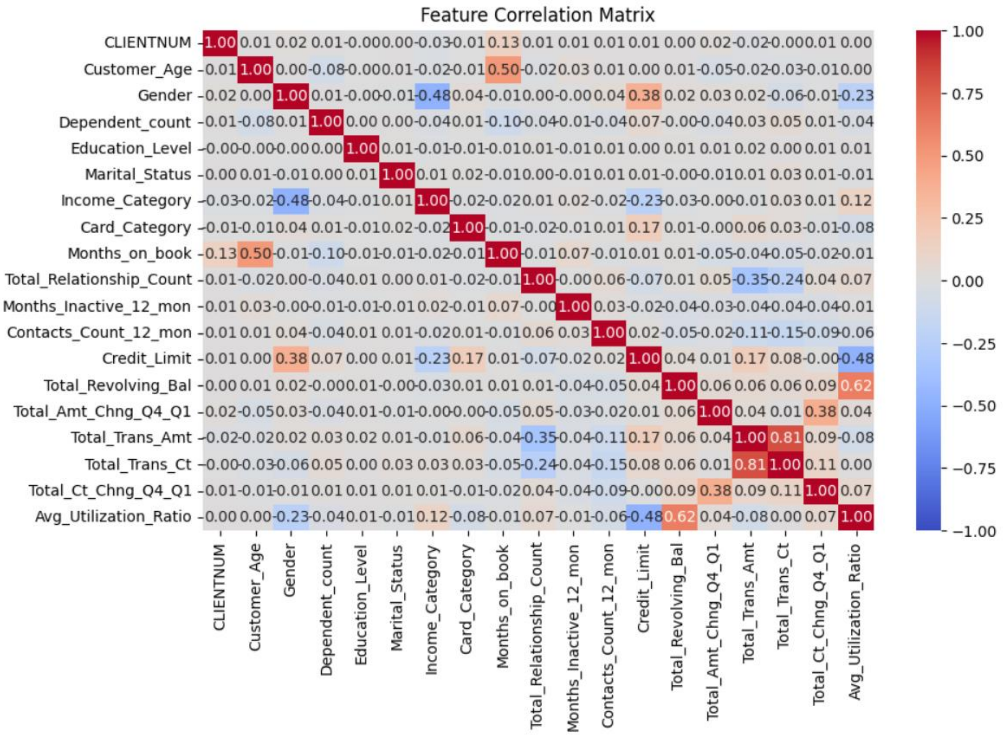
در بعضی اطلاعات داده شده به وضوح دیتاهای پرت مشاهده می‌شود. برای مثال در ویژگی سن افراد به طور واضحی قابل مشاهده است که، در این دیتاست فردی با سن 352 وجود دارد!

در مرحله بعد دیتاهای تکراری بر اساس **CLIENTNUM** سنجیده شده، و مقادیر تکراری حذف شدند.

انتخاب ویژگی:

با استفاده از فانکشن dataset.corr()، ماتریس همبستگی بین تمامی جفت ویژگی‌های موجود در دیتاست محاسبه می‌شود. مقادیر همبستگی بین ۱- و ۱ قرار دارند، که ۱- نشان‌دهنده همبستگی منفی کامل، ۰ نشان‌دهنده عدم همبستگی، و ۱ نشان‌دهنده همبستگی مثبت کامل است.

در واقع قدر مطلق مقادیر نشان دهده میزان همبستگی ویژگی‌ها هستند که با بیشتر شدن این مقدار، رنگ آن تیره تر نشان داده شده.



با تحلیل مقادیر به روش گفته شده به بی اهمیت بودن بعضی ویژگی‌ها پی می‌بریم. و سپس با حذف این ویژگی‌ها، به بهبود روند train کمک می‌کنیم.

```
df = df.drop(['CLIENTNUM', 'Unnamed: 19', 'Total_Ct_Chng_Q4_Q1', 'Months_on_book', 'Education_Level', 'Marital_Status','Customer_Age','Contacts
              'Months_Inactive_12_mon'], axis=1)
```

Z_Score

نمره **(z-score)** z یک مقدار استاندارد شده است که نشان می‌دهد یک داده چقدر از میانگین دور است، به واحد انحراف معیار. این مقدار به شما اجازه می‌دهد تا تشخیص دهید یک داده تا چه اندازه نسبت به میانگین داده‌های یک مجموعه داده‌ها پرت است. نمره z به صورت زیر محاسبه می‌شود:

$$Z = \frac{x - \mu}{\sigma}$$

در مرحله بعد به منظور شناسایی و حذف داده‌های پرت **(outliers)** در یک دیتاست نمره **(z-score)** z محاسبه شده است. داده‌های پرت در اینجا به داده‌هایی اشاره دارد که مقدارشان بیش از ۳ انحراف معیار از میانگین فاصله دارند.

در نهایت، دیتاست پس از حذف داده‌های پرت برای دو ویژگی **Total_Trans_Amt** و **Total_Trans_Ct** باقی می‌ماند. این فرآیند به بهبود کیفیت داده‌ها و کاهش تأثیرات منفی داده‌های پرت بر روی تحلیل‌ها و مدل‌های بعدی کمک می‌کند.

مقداردهی مقادیر مفقود:

روش های مختلفی جهت انجام این مرحله صورت گرفت، مثل دراپ کردن، میانگین برای ویژگی های عددی و گرفتن مد برای ویژگی های دسته ای. سپس جهت کسب نتیجه بهتر از الگوریتم هایی برای پیش‌بینی این مقادیر استفاده شد.

Mean:

```
# dataset = dataset.dropna()
dataset['Months_on_book'].fillna(value=dataset['Months_on_book'].mean(), inplace=True)
dataset['Total_Relationship_Count'].fillna(value=dataset['Total_Relationship_Count'].mean(), inplace=True)
```

K-Means:

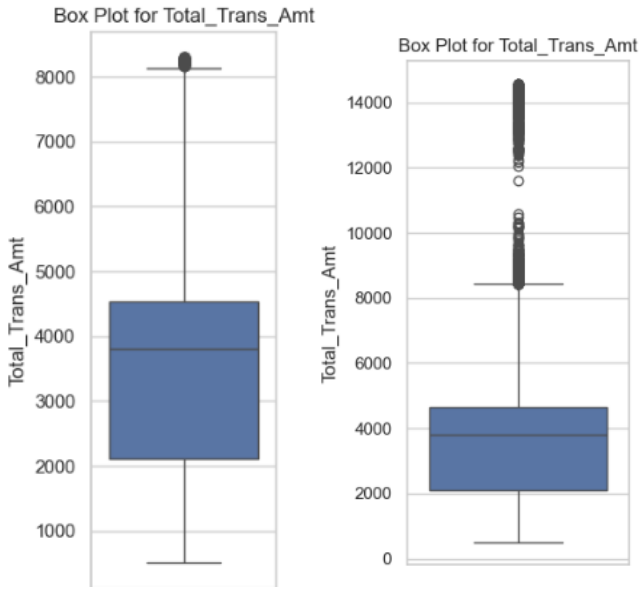
```
#first with mean
imputer = SimpleImputer(strategy='mean')
data_imputed = imputer.fit_transform(df)

#convert to dataFrame
df_imputed = pd.DataFrame(data_imputed, columns=df.columns)

#k_means to predict lost variables
kmeans = KMeans(n_clusters=10, random_state=42)
clusters = kmeans.fit_predict(data_imputed)

#adding a column for clusters
df_imputed['cluster'] = clusters


for feature in df.columns:
    if df[feature].isnull().any():
        for cluster in range(kmeans.n_clusters):
            # set the mean
            cluster_mean = df_imputed[df_imputed['cluster'] == cluster][feature].mean()

            #replace mean with the lost one
            df_imputed.loc[(df_imputed['cluster'] == cluster) & (df[feature].isnull()), feature] = cluster_mean

#drop added column
df_final = df_imputed.drop(columns=['cluster'])
```

KNN:

```
imputer = KNNImputer(n_neighbors=9)
df_knn = imputer.fit_transform(df)
knn_filled = pd.DataFrame(df_knn, columns=df.columns)
df = knn_filled
```

اما در نهایت مشاهده شد که **KNN** بهترین نتیجه را خواهد داشت.

در این مرحله به منظور شناسایی توزیع داده ها و مقادیر پرت، نمودارهای جعبه‌ای **(Box Plot)** برای تمامی ستون‌های موجود در دیتاست، رسم شد.



برای این منظور با استفاده از محدوده بین چارکی **(IQR)** استفاده شده است. به طور خاص، این کد از روش IQR برای تعیین مرزهای پایین و بالا استفاده می‌کند و سپس مقادیری را که خارج از این مرزها هستند حذف می‌کند.

چارک اول و سوم را محاسبه می‌کند.که این خطوط برای چارک اول (۲۵ درصد پایین) و چارک سوم (۷۵ درصد پایین) داده‌های ستون جاری را در نظر می‌گیرد. و سپس به محاسبه **IQR** محدوده بین چارکی پرداخته شده. سپس مرزهای پایین و بالا برای شناسایی مقادیر پرت محاسبه شد. هر مقداری که پایین‌تر از **lower_bound** یا بالاتر از **upper_bound** باشد، به عنوان مقدار پرت در نظر گرفته می‌شود. پس اقدام به حذف آن می‌کنیم.

مراحل آموزش مدل:

در این مرحله به جداسازی ستون **credit limit** از ویژگی های دیگر، به عنوان **lable** می‌پردازیم. و تمامی ویژگی های باقی مانده را به عنوان فیچر در نظر می‌گیریم.

پس از آن با تقسیم داده ها به دو فاز **train** و **test** می‌پردازیم. این کار با **random_state** مشخص، به منظور ثابت بودن تقسیم‌بندی جهت مقایسه مقادیر انجام شده. مقدار **test_size**، 0.2 در نظر گرفته شده. یعنی 20 درصد داده ها به فاز **test** و سایرین به فاز **train** تخصیص داده شوند.

```
train_pool = Pool(data=X_train, label=y_train, cat_features=categorical_features_indices)
test_pool = Pool(data=X_test, label=y_test, cat_features=categorical_features_indices)
```

این کد مربوط به استفاده از کتابخانه **CatBoost** برای ایجاد مجموعه‌های داده‌های آموزشی و آزمایشی است.

CatBoost یکی از الگوریتم‌های یادگیری ماشین است که به طور خاص برای کار با داده‌های دسته‌ای **(categorical data)** بهینه شده است. این کتابخانه توسط **Yandex** توسعه داده شده و برای بسیاری از مسائل یادگیری نظارت‌شده **(supervised learning)** استفاده می‌شود.

```
# Train the CatBoost model
model = CatBoostRegressor(
    iterations=10000,
    learning_rate=0.1,
    eval_metric='RMSE',
    logging_level='Verbose',
    use_best_model=True, nan_mode='Max')
```

در اینجا مدل **catBoost** با پارامترهای مختلف ایجاد شده:

- تعداد تکرارها برای آموزش مدل در اینجا، **10000** تکرار انتخاب شده است.
- نرخ یادگیری برای آموزش مدل، تنظیم می‌کند که هر تکرار چقدر باید مدل را بهبود دهد.
- معیار ارزیابی مدل. در اینجا، معیار **(Root Mean Squared Error)** RMSE انتخاب شده است.

- به سطح گزارش‌دهی در طول آموزش، مقدار Verbos داده شده. بدین معنا که جزئیات آموزش به طور کامل نمایش داده شود.
- مقادیر **nun** با حداکثر مقدار ممکن جایگزین می‌شوند.
- مقداردهی بولین **use best model**، **True** در نظر گرفته شده به آن معناست که، بهترین مدل که در طول آموزش مشاهده شده را استفاده می‌کند. این پارامتر اطمینان می‌دهد که بهترین مدل مشاهده شده در طول آموزش برای پیش‌بینی‌ها استفاده شود.

```
# Train the model with evaluation
model.fit(
    train_pool,
    eval_set=test_pool,
     # Stops if validation RMSE doesn't improve for 50 rounds
    early_stopping_rounds=50)
```

مقدار دهی **early_stopping** بدان معناست که اگر معیار ارزیابی در طول مشخصی تکرار بهبود نیابد، آموزش متوقف می‌شود. که در اینجا ۵۰ در نظر گرفته شده. این پارامتر کمک می‌کند تا مدل در صورتی که بهبود بیشتری در معیار ارزیابی مشاهده نشود، به زودی متوقف شود و از **overfitting** جلوگیری کند.

گزارشات حین آموزش:

```
69:     learn: 1473.3149538     test: 1603.4347116     best: 1603.4347116 (69) total: 96.1ms   remaining: 13.6s
70:     learn: 1470.3029642     test: 1604.0674952     best: 1603.4347116 (69) total: 97.5ms   remaining: 13.6s
71:     learn: 1468.9278092     test: 1602.4901292     best: 1602.4901292 (71) total: 98.8ms   remaining: 13.6s
72:     learn: 1467.7621301     test: 1601.6199263     best: 1601.6199263 (72) total: 100ms    remaining: 13.6s
73:     learn: 1466.5006263     test: 1600.2479025     best: 1600.2479025 (73) total: 101ms    remaining: 13.6s
74:     learn: 1464.7523379     test: 1597.7409129     best: 1597.7409129 (74) total: 103ms    remaining: 13.6s
75:     learn: 1462.2648243     test: 1594.2352305     best: 1594.2352305 (75) total: 104ms    remaining: 13.6s
76:     learn: 1459.3362607     test: 1593.4234039     best: 1593.4234039 (76) total: 106ms    remaining: 13.6s
77:     learn: 1458.6882044     test: 1593.2924488     best: 1593.2924488 (77) total: 107ms    remaining: 13.6s
78:     learn: 1456.9829085     test: 1593.1486083     best: 1593.1486083 (78) total: 108ms    remaining: 13.6s
79:     learn: 1454.2294206     test: 1593.0407184     best: 1593.0407184 (79) total: 109ms    remaining: 13.6s
80:     learn: 1453.1164471     test: 1592.6628730     best: 1592.6628730 (80) total: 111ms    remaining: 13.6s
81:     learn: 1451.0337310     test: 1592.2205196     best: 1592.2205196 (81) total: 112ms    remaining: 13.6s
82:     learn: 1447.7064372     test: 1591.4577266     best: 1591.4577266 (82) total: 113ms    remaining: 13.5s
83:     learn: 1445.6859273     test: 1592.1088810     best: 1591.4577266 (82) total: 115ms    remaining: 13.5s
84:     learn: 1444.5181087     test: 1591.1788882     best: 1591.1788882 (84) total: 116ms    remaining: 13.5s
85:     learn: 1441.9680079     test: 1591.3974946     best: 1591.1788882 (84) total: 117ms    remaining: 13.5s
```

نتایج نهایی به دست آمده از الگوریتم CatBoostRegressor:

مقادیر متفاوتی برای تعداد داده های متفاوتی به دست می‌آید.

```python
# Make predictions on the test set
from sklearn.metrics import mean_squared_error, r2_score
y_pred = model.predict(test_pool)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
dataset_count = df.shape[0]

print("-----CatBoostRegressor Evaluation------\n")
print(f'Mean squared Error (MSE): {mse:.2f}')
print(f'R2 Score: {r2:.4f}')
print(f'dataset count: {dataset_count}\n')
print("----------------------------------------")
```

```
-----CatBoostRegressor Evaluation------

Mean squared Error (MSE): 1059966.70
R2 Score: 0.7903
dataset count: 6606

----------------------------------------
```

```python
# Make predictions on the test set
from sklearn.metrics import mean_squared_error, r2_score
y_pred = model.predict(test_pool)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
dataset_count = df.shape[0]

print("-----CatBoostRegressor Evaluation------\n")
print(f'Mean squared Error (MSE): {mse:.2f}')
print(f'R2 Score: {r2:.4f}')
print(f'dataset count: {dataset_count}\n')
print("----------------------------------------")
```

```
-----CatBoostRegressor Evaluation------

Mean squared Error (MSE): 975809.05
R2 Score: 0.7735
dataset count: 6214

----------------------------------------
```

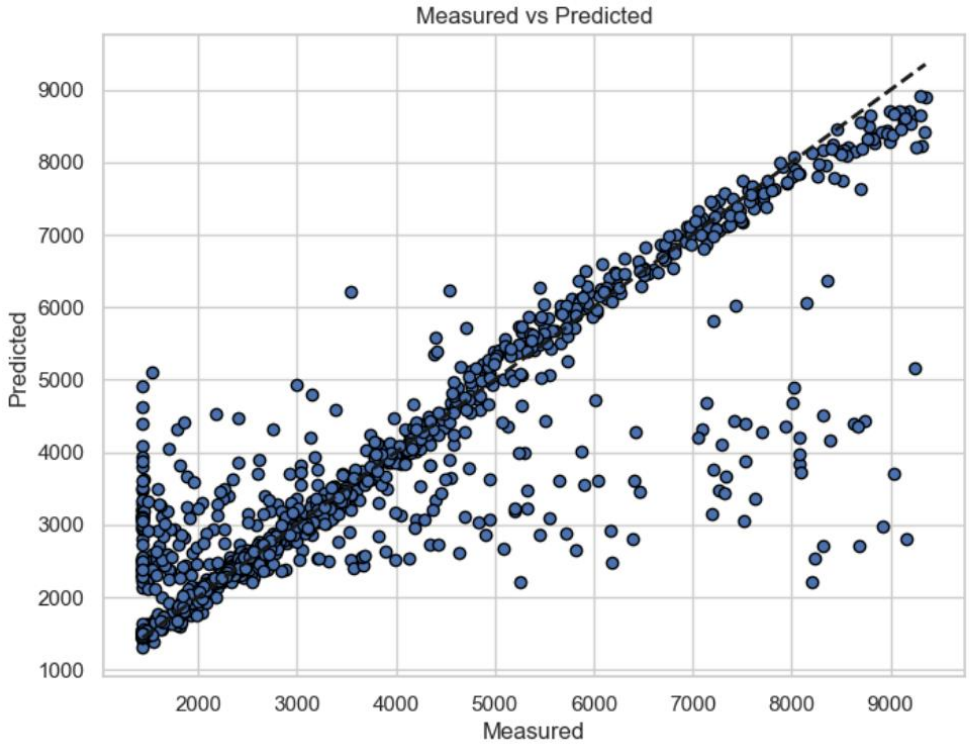
```
# Make predictions on the test set
from sklearn.metrics import mean_squared_error, r2_score
y_pred = model.predict(test_pool)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
dataset_count = df.shape[0]

print("-----CatBoostRegressor Evaluation------\n")
print(f'Mean squared Error (MSE): {mse:.2f}')
print(f'R2 Score: {r2:.4f}')
print(f'dataset count: {dataset_count}\n')
print("---------------------------------------")
```

```
-----CatBoostRegressor Evaluation------

Mean squared Error (MSE): 743391.45
R2 Score: 0.7601
dataset count: 5481

---------------------------------------
```

```
# Make predictions on the test set
from sklearn.metrics import mean_squared_error, r2_score
y_pred = model.predict(test_pool)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
dataset_count = df.shape[0]

print("-----CatBoostRegressor Evaluation------\n")
print(f'Mean squared Error (MSE): {mse:.2f}')
print(f'R2 Score: {r2:.4f}')
print(f'dataset count: {dataset_count}\n')
print("---------------------------------------")
```

```
-----CatBoostRegressor Evaluation------

Mean squared Error (MSE): 381436.70
R2 Score: 0.7631
dataset count: 3908

---------------------------------------
```

خط فیت شده با استفاده از مدل **train** شده:



قبل از انتخاب الگوریتم **catBoost** به عنوان بهترین نتیجه، انواع الگوریتم‌های **Regression** برای این دیتاست بررسی شد.

در فایل های دیگر از انواع این الگوریتم‌ها استفاده شد.

نتایج یک مدل شبکه عصبی با استفاده از **tensorflow**:

```
567/567 ━━━━━━━━━━━━━━━━━━━━ 1s 845us/step - loss: 15839223.0000 - mean_squared_error: 15839223.0000 - val_loss: 17881762.0000 - val_mean_squared_error: 17881762.0000
Epoch 293/300
567/567 ━━━━━━━━━━━━━━━━━━━━ 0s 835us/step - loss: 15101221.0000 - mean_squared_error: 15101221.0000 - val_loss: 17692108.0000 - val_mean_squared_error: 17692108.0000
Epoch 294/300
567/567 ━━━━━━━━━━━━━━━━━━━━ 1s 893us/step - loss: 17358784.0000 - mean_squared_error: 17358784.0000 - val_loss: 17621784.0000 - val_mean_squared_error: 17621784.0000
Epoch 295/300
567/567 ━━━━━━━━━━━━━━━━━━━━ 0s 789us/step - loss: 16956532.0000 - mean_squared_error: 16956532.0000 - val_loss: 17661512.0000 - val_mean_squared_error: 17661512.0000
Epoch 296/300
567/567 ━━━━━━━━━━━━━━━━━━━━ 1s 871us/step - loss: 16514942.0000 - mean_squared_error: 16514942.0000 - val_loss: 17437284.0000 - val_mean_squared_error: 17437284.0000
Epoch 297/300
567/567 ━━━━━━━━━━━━━━━━━━━━ 1s 855us/step - loss: 15935846.0000 - mean_squared_error: 15935846.0000 - val_loss: 17466248.0000 - val_mean_squared_error: 17466248.0000
Epoch 298/300
567/567 ━━━━━━━━━━━━━━━━━━━━ 0s 833us/step - loss: 16344953.0000 - mean_squared_error: 16344953.0000 - val_loss: 17450306.0000 - val_mean_squared_error: 17450306.0000
Epoch 299/300
567/567 ━━━━━━━━━━━━━━━━━━━━ 0s 798us/step - loss: 16795528.0000 - mean_squared_error: 16795528.0000 - val_loss: 17517508.0000 - val_mean_squared_error: 17517508.0000
Epoch 300/300
567/567 ━━━━━━━━━━━━━━━━━━━━ 1s 979us/step - loss: 16542008.0000 - mean_squared_error: 16542008.0000 - val_loss: 17633094.0000 - val_mean_squared_error: 17633094.0000
<keras.src.callbacks.history.History at 0x1532ad80850>
```

نتایج یک مدل Linear Regression:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error

model = LinearRegression().fit(x_train, y_train)
score = model.score(x_test,y_test)
y_pred = model.predict(x_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("-----LinearRegressor Evaluation------\n")
print(f'Mean squared Error (MSE): {mse:.2f}')
print(f'R2 Score: {r2:.4f}\n')
print("---------------------------------------")
```

```
-----LinearRegressor Evaluation------

Mean squared Error (MSE): 42535249.37
R2 Score: 0.4832

---------------------------------------
```

نتایج یک مدل Gradient Boosting Regression:

[52]:
```
GradientBoostingRegressor(max_depth=5, n_estimators=400)
```

[53]:
```
clf.score(x_test,y_test)
```

[53]:
```
0.8832770626880028
```

[54]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("-----LinearRegressor Evaluation------\n")
print(f'Mean squared Error (MSE): {mse:.2f}')
print(f'R2 Score: {r2:.4f}\n')
print("---------------------------------------")
```

```
-----LinearRegressor Evaluation------

Mean squared Error (MSE): 42535249.37
R2 Score: 0.4832

---------------------------------------
```

نتایج یک مدل Polynomial Regression:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

poly = PolynomialFeatures(degree=3)
x_train_poly = poly.fit_transform(x_train)
x_test_poly = poly.transform(x_test)

model = LinearRegression()
model.fit(x_train_poly, y_train)
y_pred = model.predict(x_test_poly)
```

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("-----LinearRegressor Evaluation------\n")
print(f'Mean squared Error (MSE): {mse:.2f}')
print(f'R2 Score: {r2:.4f}\n')
print("---------------------------------------")
```

```
-----LinearRegressor Evaluation------

Mean squared Error (MSE): 19312367.30
R2 Score: 0.7653

---------------------------------------
```

نتایج یک مدل Ridge Regression:

```
Ridge
Ridge(alpha=0.1)
```

```
mse = mean_squared_error(y_test, clf.predict(x_test))
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, clf.predict(x_test))
r2 = r2_score(y_test, y_pred)

print("-----LinearRegressor Evaluation------\n")
print(f'Mean squared Error (MSE): {mse:.2f}')
print(f'R2 Score: {r2:.4f}\n')
print("---------------------------------------")
```

```
-----LinearRegressor Evaluation------

Mean squared Error (MSE): 42535260.67
R2 Score: 0.7653

---------------------------------------
```

نتایج یک مدل RandomForest Regression:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error

model = RandomForestRegressor(max_depth=10).fit(x_train, y_train)

mse = mean_squared_error(y_test, model.predict(x_test))
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, model.predict(x_test))
r2 = r2_score(y_test, y_pred)

print("-----LinearRegressor Evaluation------\n")
print(f'Mean squared Error (MSE): {mse:.2f}')
print(f'R2 Score: {r2:.4f}\n')
print("---------------------------------------")
```

```
-----LinearRegressor Evaluation------

Mean squared Error (MSE): 10679719.90
R2 Score: 0.7653

---------------------------------------
```