400061001 ساره نقی امیری

مقدمه

در این پروژه قرار است به پیش‌بینی مشتری‌ها بر اساس اطلاعات شخصی و مالی آنها بپردازیم. هدف این پروژه این است که بتوانیم با استفاده از الگوریتم‌های یادگیری ماشین، احتمال بازپرداخت وام توسط مشتریان را پیش‌بینی کنیم. این پیش‌بینی می‌تواند به بانک‌ها و مؤسسات مالی کمک بسیار زیادی در تصمیم‌گیری درست جهت پذیرش یا رد درخواست وام کند. در این مسئله، داده‌های مشتریان شامل ویژگی‌هایی مانند سن، تحصیلات، وضعیت تأهل، درآمد ماهیانه، سابقه کار، مبلغ وام درخواستی و سابقه بازپرداخت وام‌های قبلی است. هدف ما این است که با تحلیل این داده‌ها و آموزش مدل‌های مختلف یادگیری ماشین، بتوانیم مدلی ارائه دهیم که با دقت بالا پیش‌بینی کند آیا یک مشتری قادر به بازپرداخت وام خواهد بود یا خیر. این پروژه شامل مراحل پیش‌پردازش داده، تحلیل اکتشافی داده، آموزش مدل‌های مختلف و ارزیابی عملکرد مدل‌ها است. در ادامه، ابتدا داده‌های مورد استفاده معرفی می‌شوند و سپس فرایند پیاده‌سازی مدل‌ها و تحلیل نتایج به تفصیل بررسی می‌شود.

وارد کردن کتابخانه‌های مورد نیاز

در ابتدا باید همه ابزارهای لازم برای پردازش داده، تحلیل داده و آموزش مدل‌های یادگیری ماشین را وارد کنیم. در این بخش، کتابخانه‌های numpy برای محاسبات عددی، pandas برای خواندن و پردازش داده‌های جدولی، matplotlib و seaborn برای رسم نمودارهای تحلیلی و sklearn برای پیاده‌سازی مدل‌های یادگیری ماشین فراخوانی می‌شوند. این کار باعث می‌شود تمام ابزارهای مورد نیاز برای ادامه پروژه در دسترس باشد.

```
[1] import numpy as np
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_regression

%pip
```

(pd.read_csv) در پایتون، کتابخانه pandas برای خواندن فایل‌های CreditPrediction.csv فرمت .csv قرار داده شده است. در این بخش، فایل CreditPrediction.csv را با استفاده از دستور pandas.read_csv() بارگذاری می‌کنیم و داده‌های آن را در قالب DataFrame ذخیره می‌کنیم. این کار امکان دسترسی به داده‌ها و پردازش آسان آن‌ها را فراهم می‌کند.

```
[2] data = pd.read_csv('/data/CreditPrediction.csv')
type(data)
```

```
pandas.core.frame.DataFrame
```

برای مرور اجمالی داده‌ها، از دستور data.head() استفاده می‌کنیم. این دستور پنج سطر اول داده‌ها را نمایش می‌دهد تا ساختار کلی داده‌ها، نام ستون‌ها و نوع داده‌ها بررسی شود. این کار به ما کمک می‌کند تا از صحت بارگذاری داده‌ها مطمئن شویم و ویژگی‌ها را بشناسیم. در ادامه، ابتدا داده‌های ناقص بررسی می‌شوند و سپس داده‌های دسته‌ای به فرم عددی تبدیل می‌شوند تا امکان استفاده از مدل‌های یادگیری ماشین فراهم شود.

```
[3] data.head()
```

	CLIENTNUM	Customer Age	Gender	Dependent count	Education Level	Marital Status	Income Category	Card Category	Months on book	Total Relationship Count	Months Inactive 12 mon	Contacts Count 12 mon	Credit Limit	Total Revolving Bal	Total Amt Chng Q4 Q1	Total Trans Amt	Total Trans Ct	Total Ct Chng Q4 Q1	Avg Utilization Ratio
0	768805383	45.0	M	3	High School	Married	$60K -$80K	Blue	39.0	5.0	1.0	3.0	12691.0	777.0	11914.0	1.335	1.625	0.061	0.000
1	818770008	49.0	F	5	Graduate	Married	Less than $40K	Blue	44.0	6.0	1.0	2.0	8256.0	864.0	7392.0	1.541	1.632	0.105	0.077
2	713982108	51.0	M	3	Graduate	Married	$60K -$80K	Blue	36.0	4.0	1.0	0.0	3418.0	0.0	3418.0	2.594	1.404	0.000	0.000
3	769636683	40.0	F	4	High School	Married	Less than $40K	Blue	34.0	3.0	4.0	1.0	3313.0	2517.0	796.0	1.405	1.333	0.760	0.759
4	709106358	40.0	M	3	Uneducated	Married	$60K -$80K	Blue	21.0	5.0	1.0	0.0	4716.0	0.0	4716.0	2.175	0.833	0.000	0.000

پیش‌پردازش داده‌ها

پیش‌پردازش داده‌ها یکی از مراحل مهم و حیاتی در هر پروژه یادگیری ماشین است. در این بخش، ابتدا ساختار داده‌ها با استفاده از دستور data.info() بررسی می‌شود تا نوع داده‌های هر ستون و تعداد داده‌های غیرخالی مشخص شود. سپس داده‌های ناقص شناسایی و پردازش می‌شوند و در نهایت داده‌های دسته‌ای به فرم عددی تبدیل می‌شوند تا امکان استفاده از مدل‌های یادگیری ماشین فراهم شود. این مراحل به ما کمک می‌کند تا مدل‌ها با داده‌های تمیز و استاندارد آموزش ببینند و عملکرد بهتری داشته باشند.

```
[4] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10127 entries, 0 to 10126
Data columns (total 20 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   CLIENTNUM                 10127 non-null  int64
 1   Customer Age              10127 non-null  float64
 2   Gender                    10127 non-null  object
 3   Dependent count           10127 non-null  int64
 4   Education Level           10127 non-null  object
 5   Marital Status            10127 non-null  object
 6   Income Category           10127 non-null  object
 7   Card Category             9300 non-null   object
 8   Months on book            10127 non-null  int64
 9   Total Relationship Count  10127 non-null  int64
 10  Months Inactive 12 mon    10127 non-null  int64
 11  Contacts Count 12 mon     10127 non-null  int64
 12  Credit Limit              10127 non-null  int64
 13  Total Revolving Bal       10127 non-null  int64
 14  Total Amt Chng Q4 Q1      10127 non-null  int64
 15  Total Trans Amt           10127 non-null  int64
 16  Total Trans Ct            10127 non-null  int64
 17  Total Ct Chng Q4 Q1       10127 non-null  int64
 18  Avg Utilization Ratio     10127 non-null  int64
 19  Unknown                   10127 non-null  float64
dtype: int64
memory usage: 1.6+ MB
```

تمام موارد را با داده‌های ناقص انجام دهید.

```
[5] data.isnull().sum()
```

```
CLIENTNUM                 0
Customer Age              0
Gender                    0
Dependent count           0
Education Level           0
Marital Status            0
Income Category           0
Card Category           827
Months on book            0
Total Relationship Count  0
Months Inactive 12 mon    0
Contacts Count 12 mon     0
Credit Limit              0
Total Revolving Bal       0
Total Amt Chng Q4 Q1      0
Total Trans Amt           0
Total Trans Ct            0
Total Ct Chng Q4 Q1       0
Avg Utilization Ratio     0
Unknown                   0
dtype: int64
```

تمام موارد را با داده‌های ناقص انجام دهید.

```
[6] missing_percentages = data.isnull().mean()
column_to_delete = missing_percentages[missing_percentages > 0.6].index
data.drop(column_to_delete, inplace=True)
print("Unknown' If', dtype='object')
```

در این بخش، ابتدا درصد داده‌های ناقص هر ستون محاسبه می‌شود و سپس ستون‌هایی که بیش از ۶۰ درصد داده‌های آن‌ها ناقص است، حذف می‌شوند. این کار باعث می‌شود تا داده‌های باقیمانده قابل اعتمادتر باشند و مدل‌ها بر روی داده‌های با کیفیت بالاتر آموزش ببینند. در ادامه، داده‌های باقیمانده بررسی می‌شوند و داده‌های دسته‌ای به فرم عددی تبدیل می‌شوند تا امکان استفاده از مدل‌های یادگیری ماشین فراهم شود.

```
[7] data.drop(columns = ['Unknown' IN', 'CLIENTNUM'], inplace = True)
```

در این بخش، ستون‌های Unknown و CLIENTNUM از دیتاست حذف می‌شوند. ستون Unknown به دلیل نامشخص بودن و CLIENTNUM به دلیل اینکه فقط شناسه مشتری است و تأثیری بر پیش‌بینی ندارد، از مجموعه داده حذف می‌شوند. این کار باعث می‌شود تا مدل‌ها بر روی ویژگی‌های مفید و معنادار آموزش ببینند و از پیچیدگی محاسبات کاسته شود.

```
[8] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10127 entries, 0 to 10126
Data columns (total 18 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Customer Age              10127 non-null  float64
 1   Gender                    10127 non-null  object
 2   Dependent count           10127 non-null  int64
 3   Education Level           10127 non-null  object
 4   Marital Status            10127 non-null  object
 5   Income Category           10127 non-null  object
 6   Card Category             9300 non-null   object
 7   Months on book            10127 non-null  int64
 8   Total Relationship Count  10127 non-null  int64
 9   Months Inactive 12 mon    10127 non-null  int64
 10  Contacts Count 12 mon     10127 non-null  int64
 11  Credit Limit              10127 non-null  int64
 12  Total Revolving Bal       10127 non-null  int64
 13  Total Amt Chng Q4 Q1      10127 non-null  int64
 14  Total Trans Amt           10127 non-null  int64
 15  Total Trans Ct            10127 non-null  int64
 16  Total Ct Chng Q4 Q1       10127 non-null  int64
 17  Avg Utilization Ratio     10127 non-null  int64
dtypes: float64(1), int64(14), object(3)
memory usage: 1.4 MB
```

تعداد ستون‌های دسته‌ای (object) را بشمار.

```
[9] data.dtypes.value_counts()
```

```
Customer Age              float64
Gender                     object
Dependent count            int64
Education Level            object
Marital Status             object
Income Category            object
Card Category              object
Months on book             int64
Total Relationship Count   int64
Months Inactive 12 mon     int64
Contacts Count 12 mon      int64
Credit Limit               int64
Total Revolving Bal        int64
Total Amt Chng Q4 Q1       int64
Total Trans Amt            int64
Total Trans Ct             int64
Total Ct Chng Q4 Q1        int64
Avg Utilization Ratio      int64
dtype: int64
```

تعداد ستون‌های دسته‌ای (object) را بشمار.

```
[10] data.dtypes.value_counts()
```

```
Customer Age              float64
Gender                     object
Dependent count            int64
Education Level            object
Marital Status             object
Income Category            object
Card Category              object
Months on book             int64
Total Relationship Count   int64
Months Inactive 12 mon     int64
Contacts Count 12 mon      int64
Credit Limit               int64
Total Revolving Bal        int64
Total Amt Chng Q4 Q1       int64
Total Trans Amt            int64
Total Trans Ct             int64
Total Ct Chng Q4 Q1        int64
Avg Utilization Ratio      int64
dtype: int64
```

داده‌های دسته‌ای

تمام داده‌های موجود در دیتاست باید به صورت کامل عددی باشند تا بتوانیم مدل‌های یادگیری ماشین را روی آن‌ها آموزش دهیم. در این بخش، ابتدا با استفاده از دستور data.info() بررسی می‌شود تا مشخص شود کدام ستون‌ها هنوز به صورت رشته (object) باقی مانده‌اند و نیاز به تبدیل به فرم عددی دارند. این کار باعث می‌شود تا مدل‌ها بتوانند از تمام ویژگی‌ها استفاده کنند و از بروز خطا در فرایند آموزش مدل جلوگیری شود.

```
[11] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10127 entries, 0 to 10126
Data columns (total 18 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Customer Age              10127 non-null  float64
 1   Gender                    10127 non-null  object
 2   Dependent count           10127 non-null  int64
 3   Education Level           10127 non-null  object
 4   Marital Status            10127 non-null  object
 5   Income Category           10127 non-null  object
 6   Card Category             9300 non-null   object
 7   Months on book            10127 non-null  int64
 8   Total Relationship Count  10127 non-null  int64
 9   Months Inactive 12 mon    10127 non-null  int64
 10  Contacts Count 12 mon     10127 non-null  int64
 11  Credit Limit              10127 non-null  int64
 12  Total Revolving Bal       10127 non-null  int64
 13  Total Amt Chng Q4 Q1      10127 non-null  int64
 14  Total Trans Amt           10127 non-null  int64
 15  Total Trans Ct            10127 non-null  int64
 16  Total Ct Chng Q4 Q1       10127 non-null  int64
 17  Avg Utilization Ratio     10127 non-null  int64
dtypes: float64(1), int64(14), object(3)
memory usage: 1.4 MB
```

```
[12] data['Gender'].unique()
```

```
array(['M', 'F', nan], dtype=object)
```

```
[13] data['Marital Status'].unique()
```

```
array(['Married', 'Unknown', 'Single', 'Divorced', dtype=object)
```

```
[14] data = pd.get_dummies(data, columns=['Gender', 'Marital Status'], dtype=int)
```

در این بخش، ابتدا با استفاده از دستور data.info() بررسی می‌شود تا مشخص شود کدام ستون‌ها هنوز به صورت رشته (object) باقی مانده‌اند و نیاز به تبدیل به فرم عددی دارند. سپس ستون‌های Gender و Marital Status با استفاده از دستور pd.get_dummies() به صورت دوتایی (one-hot encoding) تبدیل می‌شوند تا مدل‌ها بتوانند از این ویژگی‌ها استفاده کنند و از بروز خطا در فرایند آموزش مدل جلوگیری شود.

```
[15] for column in ['Education_Level']:
    print(data[column].unique().__sep__())
```

```
['High School' 'Graduate' 'Uneducated' 'College' 'Post-Graduate'
 'Doctorate']
```

```
[16] for column in ['Income_Category']:
    print(data[column].unique().__sep__())
```

```
['$60K - $80K' 'Less than $40K' '$80K - $120K' '$40K - $60K' '$120K +']
```

```
[17] for column in ['Card_Category']:
    print(data[column].unique().__sep__())
```

```
['Blue' 'Gold' 'Silver' 'Platinum']
```

```
Education_Level_map = {'High School': 1, 'Graduate': 3, 'Uneducated': 0, 'College': 2, 'Post-Graduate': 4,
'Doctorate': 5}
data['Education_Level'] = data['Education_Level'].map(Education_Level_map)

Income_Category_map = {'$60K - $80K': 2, 'Less than $40K': 0, '$80K - $120K': 3, '$40K - $60K': 1, '$120K +': 4}
data['Income_Category'] = data['Income_Category'].map(Income_Category_map)

Card_Category_map = {'Blue': 0, 'Gold': 2, 'Silver': 1, 'Platinum': 4}
data['Card_Category'] = data['Card_Category'].map(Card_Category_map)
```

در این بخش، ابتدا با استفاده از دستور data.info() بررسی می‌شود تا مشخص شود کدام ستون‌ها هنوز به صورت رشته (object) باقی مانده‌اند و نیاز به تبدیل به فرم عددی دارند. سپس ستون‌های Education_Level، Income_Category و Card_Category با استفاده از دستور map() به صورت دستی تبدیل می‌شوند تا مدل‌ها بتوانند از این ویژگی‌ها استفاده کنند و از بروز خطا در فرایند آموزش مدل جلوگیری شود.

```
[18] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10127 entries, 0 to 10126
Data columns (total 21 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Customer Age              10127 non-null  float64
 1   Dependent count           10127 non-null  int64
 2   Education Level           10127 non-null  int64
 3   Income Category           10127 non-null  int64
 4   Card Category             9300 non-null   int64
 5   Months on book            10127 non-null  int64
 6   Total Relationship Count  10127 non-null  int64
 7   Months Inactive 12 mon    10127 non-null  int64
 8   Contacts Count 12 mon     10127 non-null  int64
 9   Credit Limit              10127 non-null  int64
 10  Total Revolving Bal       10127 non-null  int64
 11  Total Amt Chng Q4 Q1      10127 non-null  int64
 12  Total Trans Amt           10127 non-null  int64
 13  Total Trans Ct            10127 non-null  int64
 14  Total Ct Chng Q4 Q1       10127 non-null  int64
 15  Avg Utilization Ratio     10127 non-null  int64
 16  Gender_F                  10127 non-null  int64
 17  Gender_M                  10127 non-null  int64
 18  Marital Status_Divorced   10127 non-null  int64
 19  Marital Status_Married    10127 non-null  int64
 20  Marital Status_Single     10127 non-null  int64
dtypes: float64(1), int64(20)
memory usage: 1.6 MB
```

حذف داده‌های پرت

در این بخش، ابتدا با استفاده از دستور data.info() بررسی می‌شود تا مشخص شود کدام ستون‌ها هنوز به صورت رشته (object) باقی مانده‌اند و نیاز به تبدیل به فرم عددی دارند. سپس ستون‌های Gender و Marital Status با استفاده از دستور pd.get_dummies() به صورت دوتایی (one-hot encoding) تبدیل می‌شوند تا مدل‌ها بتوانند از این ویژگی‌ها استفاده کنند و از بروز خطا در فرایند آموزش مدل جلوگیری شود.

```
[19] iqr = IQR
count = np.count_nonzero(data['Customer_Age'] > iqr)
print("Count of values greater than {iqr} in Column Customer_Age: {count}")
count = np.count_nonzero(data['Customer_Age'] < -iqr)
print("Count of values greater than 100 in Column Customer_Age: {count}")
data = data[data['Customer_Age'] < iqr]
```

با استفاده از دستور data.info() بررسی می‌شود تا مشخص شود کدام ستون‌ها هنوز به صورت رشته (object) باقی مانده‌اند و نیاز به تبدیل به فرم عددی دارند. سپس ستون‌های Gender و Marital Status با استفاده از دستور pd.get_dummies() به صورت دوتایی (one-hot encoding) تبدیل می‌شوند تا مدل‌ها بتوانند از این ویژگی‌ها استفاده کنند و از بروز خطا در فرایند آموزش مدل جلوگیری شود.

```
[20] data.dtypes
```

```
Customer_Age              float64
Dependent count            int64
Education Level            int64
Income Category            int64
Card Category              int64
Months on book             int64
Total Relationship Count   int64
Months Inactive 12 mon     int64
Contacts Count 12 mon      int64
Credit Limit               int64
Total Revolving Bal        int64
Total Amt Chng Q4 Q1       int64
Total Trans Amt            int64
Total Trans Ct             int64
Total Ct Chng Q4 Q1        int64
Avg Utilization Ratio      int64
Gender_F                   int64
Gender_M                   int64
Marital Status_Divorced    int64
Marital Status_Married     int64
Marital Status_Single      int64
dtype: float64(1), int64(20)
memory usage: 1.5 MB
```

توجه کنید و استاد کردن داده‌ها

بعضی از ستون‌ها مقدار بسیار کمی دارند و مدل‌ها نمی‌توانند از آن‌ها به درستی استفاده کنند. در این بخش، ابتدا با استفاده از دستور data.info() بررسی می‌شود تا مشخص شود کدام ستون‌ها دارای مقدار بسیار کم هستند. سپس ستون‌های Gender_F و Gender_M با استفاده از دستور drop() حذف می‌شوند تا مدل‌ها بتوانند از داده‌های متنوع‌تر و مناسب‌تر استفاده کنند و از بروز خطا در فرایند آموزش مدل جلوگیری شود.

```
[21] y = data['Credit_Limit']
data.drop(columns = ['Credit_Limit'], inplace = True)
```

در این بخش، ابتدا با استفاده از دستور data.info() بررسی می‌شود تا مشخص شود کدام ستون‌ها دارای مقدار بسیار کم هستند. سپس ستون‌های Gender_F و Gender_M با استفاده از دستور drop() حذف می‌شوند تا مدل‌ها بتوانند از داده‌های متنوع‌تر و مناسب‌تر استفاده کنند و از بروز خطا در فرایند آموزش مدل جلوگیری شود.

```
[22] data.dtypes
```

```
Customer_Age              float64
Dependent count            int64
Education Level            int64
Income Category            int64
Card Category              int64
Months on book             int64
Total Relationship Count   int64
Months Inactive 12 mon     int64
Contacts Count 12 mon      int64
Total Revolving Bal        int64
Total Amt Chng Q4 Q1       int64
Total Trans Amt            int64
Total Trans Ct             int64
Total Ct Chng Q4 Q1        int64
Avg Utilization Ratio      int64
Gender_F                   int64
Gender_M                   int64
Marital Status_Divorced    int64
Marital Status_Married     int64
Marital Status_Single      int64
dtype: float64(1), int64(20)
memory usage: 1.5 MB
```

ساخت مدل

در این بخش، ابتدا با استفاده از دستور data.info() بررسی می‌شود تا مشخص شود کدام ستون‌ها دارای مقدار بسیار کم هستند. سپس ستون‌های Gender_F و Gender_M با استفاده از دستور drop() حذف می‌شوند تا مدل‌ها بتوانند از داده‌های متنوع‌تر و مناسب‌تر استفاده کنند و از بروز خطا در فرایند آموزش مدل جلوگیری شود.

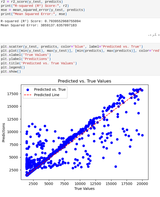
```
[23] X_train, X_test, y_train, y_test = train_test_split(data, y, test_size=0.2, shuffle=True)
```

در این بخش، ابتدا با استفاده از دستور train_test_split() داده‌ها به دو بخش آموزش (train) و آزمون (test) تقسیم می‌شوند. سپس داده‌های آموزش و آزمون به صورت تصادفی (shuffle=True) جابه‌جا می‌شوند تا مدل‌ها بتوانند از داده‌های متنوع‌تر و مناسب‌تر استفاده کنند و از بروز خطا در فرایند آموزش مدل جلوگیری شود.

```
[24] scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

[25] rf = RandomForestRegressor(n_estimators=100, max_depth=10, random_state=10)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("R-squared: {r2:.4f}")
RMSE: {rmse:.2f}
R-squared: 0.9997
RMSE: 43.09
```

در این بخش، ابتدا با استفاده از دستور StandardScaler() داده‌های آموزش و آزمون مقیاس‌بندی می‌شوند تا مدل‌ها بتوانند از داده‌های استاندارد شده استفاده کنند و از بروز خطا در فرایند آموزش مدل جلوگیری شود.

```
[26] from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

[27] rf = RandomForestRegressor(n_estimators=100, max_depth=10, random_state=10)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("R-squared: {r2:.4f}")
RMSE: {rmse:.2f}
R-squared: 0.9997
RMSE: 43.09
```

در این بخش، ابتدا با استفاده از دستور RandomForestRegressor() یک مدل جنگل تصادفی (Random Forest) با ۱۰۰ درخت (n_estimators=100) و عمق حداکثر ۱۰ (max_depth=10) ساخته می‌شود. سپس داده‌های آموزش (X_train, y_train) به مدل داده می‌شوند تا مدل آموزش ببیند. در ادامه، مدل آموزش‌دیده بر روی داده‌های آزمون (X_test) اجرا می‌شود و پیش‌بینی‌ها (y_pred) ذخیره می‌شوند. در نهایت، با استفاده از دستور mean_squared_error() و np.sqrt() مقدار خطای مدل (RMSE) محاسبه می‌شود تا میزان دقت مدل در پیش‌بینی مقادیر اعتبار مشتری ارزیابی شود.

```
[28] y_pred = rf.predict(X_test)
print("Predicted vs. True Values")
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2, label='Predicted Line')
plt.xlabel('True Values')
plt.ylabel('Predicted Values')
plt.title('Predicted vs. True Values')
plt.legend()
plt.grid()
plt.show()
```