Task

Payment_data.csv:

payment_data.csv: customer's card payment history. id: customer id, OVD_t1: number of times overdue type 1, OVD_t2: number of times overdue type 2, OVD_t3: number of times overdue type 3, OVD_sum: total overdue days, pay_normal: number of times normal payment, prod_code: credit product code, prod_limit: credit limit of product, update_date: account update date, new_balance: current balance of product, highest_balance: highest balance in history, report_date: date of recent payment,

Customer_data.csv:

customer's demographic data and category attributes which have been encoded. Category features are fea_1, fea_3, fea_5, fea_6, fea_7, fea_9. label is 1, the customer is in high credit risk and label is 0, the customer is in low credit risk

When we consider the explanation of customer and payment data variables we conclude that these data sets give us overdue type and total overdue days, normal payment number and credit detail thus our target variable is label because when i think as a worker in the bank number of overdue type other variables can not target variable they can be features because bank earn money from credit and they want to give credit to reliable people so, banks pay close attention to the credit risk of customers

In [40]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.linear_model import LogisticRegressionCV
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve,roc_auc_score, RocCurveDisplay
```

In [41]:
```
# Veri setlerinin tanımlanması
payments = pd.read_csv("payment_data.csv")
payments = payments.set_index("id")

customers = pd.read_csv("customer_data.csv")
customers = customers.set_index("id")

# İki veri setinin birleştirilmesi
customer_data = customers.join(payments)
customer_data
```

Out[41]:

	label	fea_1	fea_2	fea_3	fea_4	fea_5	fea_6	fea_7	fea_8	fea_9	...	OVD_t
id												
54982353	0	1	1130.0	2	1000000.0	2	4	-1	100	5	...	(
54982353	0	1	1130.0	2	1000000.0	2	4	-1	100	5	...	(
54982353	0	1	1130.0	2	1000000.0	2	4	-1	100	5	...	(
54982353	0	1	1130.0	2	1000000.0	2	4	-1	100	5	...	(
54982353	0	1	1130.0	2	1000000.0	2	4	-1	100	5	...	(
...	...	...	...	...	...	...	...	...	...	...	...	.
59006219	0	4	NaN	2	111000.0	2	8	5	110	4	...	(
59006219	0	4	NaN	2	111000.0	2	8	5	110	4	...	(
59006239	0	7	1322.0	3	68000.0	2	11	5	86	3	...	(
59006239	0	7	1322.0	3	68000.0	2	11	5	86	3	...	(
59006239	0	7	1322.0	3	68000.0	2	11	5	86	3	...	(

8250 rows × 23 columns

In [42]:

```
customer_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8250 entries, 54982353 to 59006239
Data columns (total 23 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   label            8250 non-null   int64  
 1   fea_1            8250 non-null   int64  
 2   fea_2            7222 non-null   float64
 3   fea_3            8250 non-null   int64  
 4   fea_4            8250 non-null   float64
 5   fea_5            8250 non-null   int64  
 6   fea_6            8250 non-null   int64  
 7   fea_7            8250 non-null   int64  
 8   fea_8            8250 non-null   int64  
 9   fea_9            8250 non-null   int64  
 10  fea_10           8250 non-null   int64  
 11  fea_11           8250 non-null   float64
 12  OVD_t1           8250 non-null   int64  
 13  OVD_t2           8250 non-null   int64  
 14  OVD_t3           8250 non-null   int64  
 15  OVD_sum          8250 non-null   int64  
 16  pay_normal       8250 non-null   int64  
 17  prod_code        8250 non-null   int64  
 18  prod_limit       2132 non-null   float64
 19  update_date      8224 non-null   object 
 20  new_balance      8250 non-null   float64
 21  highest_balance  7841 non-null   float64
 22  report_date      7136 non-null   object 
dtypes: float64(6), int64(15), object(2)
memory usage: 1.5+ MB
```

In [43]:
```
customer_data.shape
```

Out[43]:
```
(8250, 23)
```

There is 8250 observations and 23 variables and thersis 6 float, 15 int64 and 2 objects then we look at these objects and can we turn into int64? These two objects are date and we are not working time series so we should remove from data.

In [44]:
```
customer_data.isnull().sum()
```

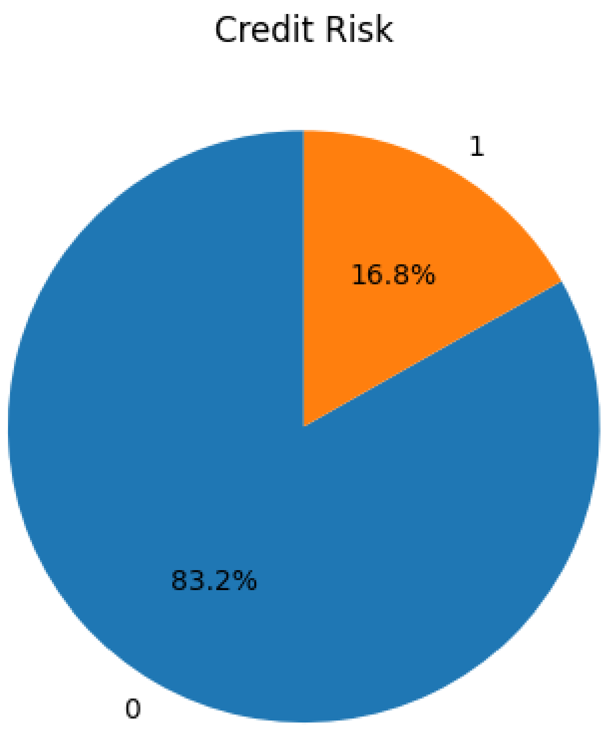
Out[44]:
```
label                 0
fea_1                 0
fea_2              1028
fea_3                 0
fea_4                 0
fea_5                 0
fea_6                 0
fea_7                 0
fea_8                 0
fea_9                 0
fea_10                0
fea_11                0
OVD_t1                0
OVD_t2                0
OVD_t3                0
OVD_sum               0
pay_normal            0
prod_code             0
prod_limit         6118
update_date          26
new_balance           0
highest_balance     409
report_date        1114
dtype: int64
```

fae_2, prod_limit and highest_balance have too much null so ı remove this variable from dataset. because if we delete as rows we lost so many data from other data.

In [45]:
```
cd = customer_data.drop(['fea_2','prod_limit','update_date','report_date','highe
```

Trianing a Logistic Regression Model

In [46]:
```
counts = cd['label'].value_counts()
plt.pie(counts, labels=counts.index, autopct='%1.1f%%', startangle=90)

plt.title('Credit Risk')
plt.show()
```



In [72]:
```
y = cd['label']
x = cd.drop(['label'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_
model = LogisticRegressionCV(Cs=10, cv=5, random_state=42)
result = model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print(model.score(X_test,y_test))
```

```
0.8351515151515152
```

In [70]:
```
cm = confusion_matrix(y_test, y_pred)

print(cm)
```

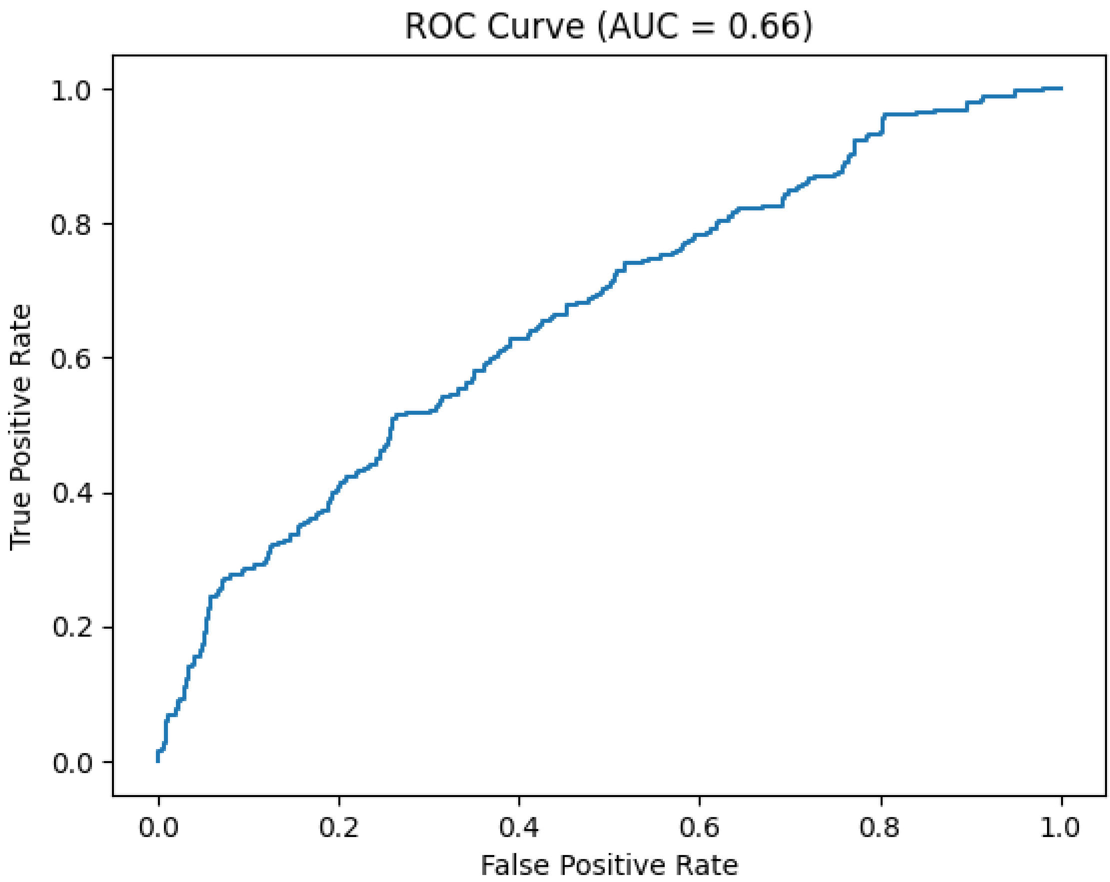
```
[[1374    2]
 [ 270    4]]
```

In [71]:
```
report = classification_report(y_test, y_pred)

print(report)
```

```
              precision    recall  f1-score   support

           0       0.84      1.00      0.91      1376
           1       0.67      0.01      0.03       274

    accuracy                           0.84      1650
   macro avg       0.75      0.51      0.47      1650
weighted avg       0.81      0.84      0.76      1650
```

In [65]:
```
fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:, 1])

roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr)

roc_auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])
```

```
fig, ax = plt.subplots()
roc_display.plot(ax=ax)

ax.set_xlabel('False Positive Rate')
ax.set_ylabel('True Positive Rate')
ax.set_title(f'ROC Curve (AUC = {roc_auc:.2f})')
plt.show()
```



Overall, our model has a good accuracy score of 0.84, but when we look at the confusion matrix, we can see that the prediction for the value 1 is almost entirely incorrect. Additionally, the AUC value is 0.66. We will try to reduce the error in predicting the value 1 by using under and over sampling, and check if we can obtain a higher AUC value

İmbalanced Problem(Under and Over Sampling)

In [52]:
```
# undersampling
rus = RandomUnderSampler(random_state=42)
X_resampled, y_resampled = rus.fit_resample(x, y)
X_trainus, X_testus, y_trainus, y_testus = train_test_split(X_resampled, y_resam
model2 = LogisticRegressionCV(Cs=10, cv=5, random_state=42)
result = model2.fit(X_trainus,y_trainus)
y_predus = model2.predict(X_testus)
print(model2.score(X_testus,y_testus))
```

```
0.6378378378378379
```

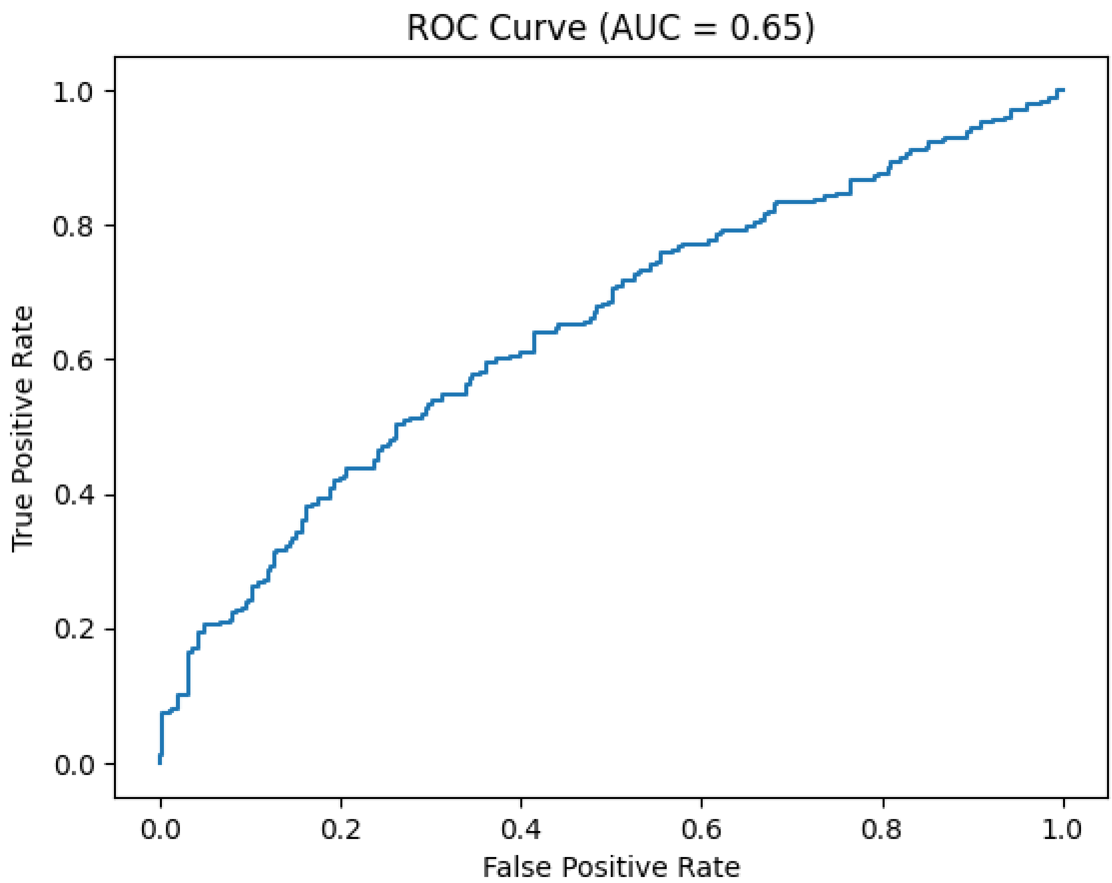
In [63]:
```
cm2 = confusion_matrix(y_testus, y_predus)

print(cm2)
```

```
[[163 122]
 [ 79 191]]
```

In [55]:
```
report = classification_report(y_testus, y_predus)

print(report)
```

```
              precision    recall  f1-score   support

           0       0.67      0.57      0.62       285
           1       0.61      0.71      0.66       270

    accuracy                           0.64       555
   macro avg       0.64      0.64      0.64       555
weighted avg       0.64      0.64      0.64       555
```

In [67]:
```
fpr, tpr, _ = roc_curve(y_testus, model.predict_proba(X_testus)[:, 1])

roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr)

roc_auc = roc_auc_score(y_testus, model.predict_proba(X_testus)[:, 1])

fig, ax = plt.subplots()
roc_display.plot(ax=ax)
ax.set_xlabel('False Positive Rate')
ax.set_ylabel('True Positive Rate')
ax.set_title(f'ROC Curve (AUC = {roc_auc:.2f})')
plt.show()
```



In [58]:
```
# oversampling
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(x, y)
```

```
X_trainos, X_testos, y_trainos, y_testos = train_test_split(X_resampled, y_resam
model3 = LogisticRegressionCV(Cs=10, cv=5, random_state=42)
result = model3.fit(X_trainos,y_trainos)
y_predos = model3.predict(X_testos)
print(model3.score(X_testos,y_testos))
```

```
0.6154406409322651
```

In [62]:

```
cm3 = confusion_matrix(y_testos, y_predos)

print(cm3)
```

```
[[707 669]
 [387 983]]
```

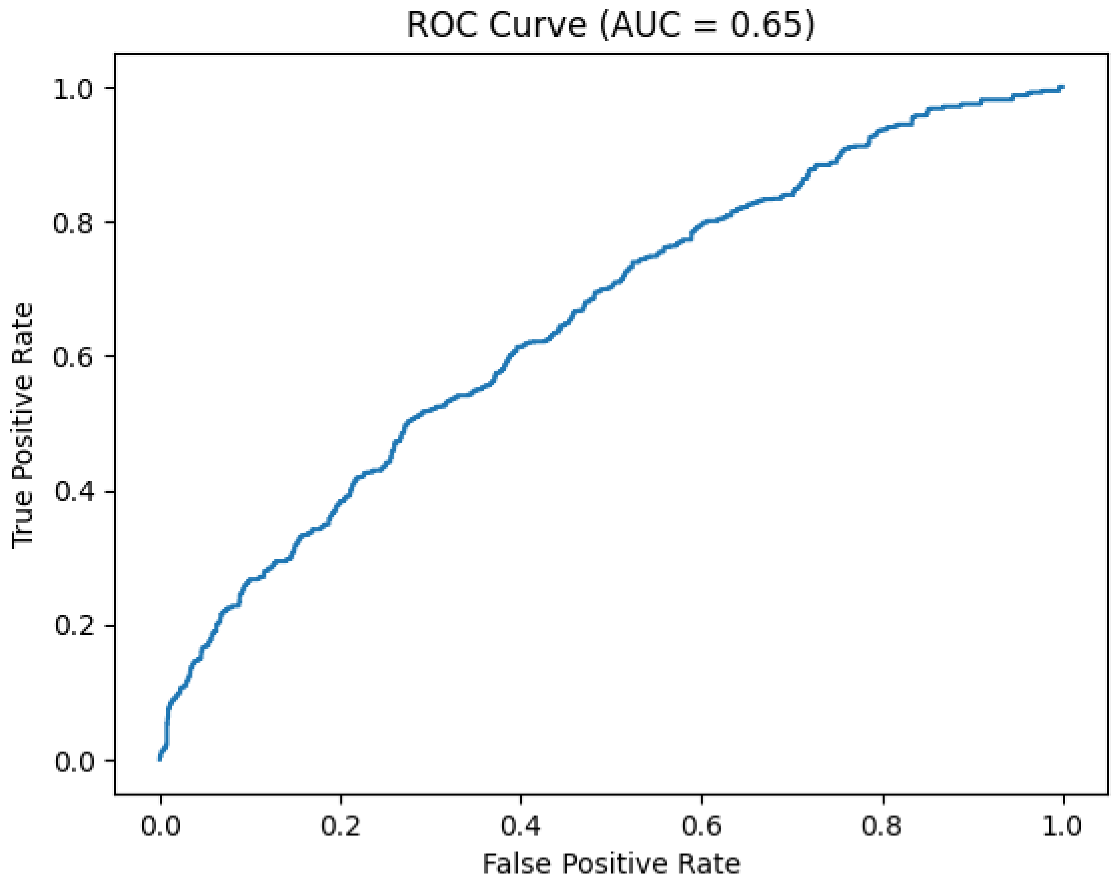
In [60]:

```
report = classification_report(y_testos, y_predos)

print(report)
```

```
              precision    recall  f1-score   support

           0       0.65      0.51      0.57      1376
           1       0.60      0.72      0.65      1370

    accuracy                           0.62      2746
   macro avg       0.62      0.62      0.61      2746
weighted avg       0.62      0.62      0.61      2746
```

In [68]:

```
fpr, tpr, _ = roc_curve(y_testos, model.predict_proba(X_testos)[:, 1])

roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr)

roc_auc = roc_auc_score(y_testos, model.predict_proba(X_testos)[:, 1])

fig, ax = plt.subplots()
roc_display.plot(ax=ax)

ax.set_xlabel('False Positive Rate')
ax.set_ylabel('True Positive Rate')
ax.set_title(f'ROC Curve (AUC = {roc_auc:.2f})')
plt.show()
```



We were able to reduce the error in predicting the value 1 through under-sampling, but our accuracy score dropped to 0.64 and the error rate in predicting the value 0 increased significantly. Additionally, the AUC value slightly decreased to 0.65. We were able to reduce the error in predicting the value 1 through over-sampling, but our accuracy score dropped to 0.62 and the error rate in predicting the value 0 increased significantly. Additionally, the AUC value slightly decreased to 0.65. Therefore, we obtained similar results with under-sampling, but when we looked at the F1 score, under-sampling gave slightly better results.

As a result, although the accuracy we obtained with logistic regression satisfies us, since it almost always returns 0 instead of 1 in the confusion matrix, we do not have a proper classification, which means we are using a classification model that cannot classify our dataset. Therefore, even though the accuracy value in under-sampling is low, we have partially solved the imbalanced problem in the model with under-sampling, as it provides a better classification compared to the initial model.