The dataset contains information about payment failures, demographic factors, credit data, payment history, and billing statements of credit card clients in Taiwan from April 2005 to September 2005.

There are 25 variables:

- ID: ID of each client
- LIMIT_BAL: Amount of credits granted in NT dollars (includes individual and family/additional credits
- SEX: Gender (1=male, 2=female)
- EDUCATION: (1=S2, 2=university, 3=high school, 4=other, 5=unknown, 6=unknown)
- MARRIAGE: Marital status (1=married, 2=single, 3=other)
- AGE: Age in years
- PAY_0: Payment status for September 2005 (-1=paid in full, 1=payment delay for one month, 2=payment delay for two months, ... 8=payment delay for eight months, 9=payment delay for nine months and on)
- PAY_2: Repayment status in August 2005 (same scale as above)
- PAY_3: Repayment status in July 2005 (same scale as above)
- PAY_4: Repayment status in June 2005 (same scale as above)
- PAY_5: Repayment status in May 2005 (same scale as above)
- PAY_6: Repayment status in April 2005 (same scale as above)
- BILL_AMT1: Billing statement amount for September 2005 (NT dollars)
- BILL_AMT2 : Billing amount for August 2005 (NT Dollars)
- BILL_AMT3: Billing statement amount for July 2005 (NT Dollars)
- BILL_AMT4: Billing statement amount for June 2005 (NT Dollars)
- BILL_AMT5: Billing amount for May 2005 (NT Dollars)
- BILL_AMT6: Billing statement amount for April 2005 (NT Dollars)
- PAY_AMT1: Previous payment amount in September 2005 (NT dollars)
- PAY_AMT2: Previous payment amount in August 2005 (NT Dollars)
- PAY_AMT3: Previous payment amount in July 2005 (NT dollars)
- PAY_AMT4: Previous payment amount in June 2005 (NT Dollars)
- PAY_AMT5: Previous payment amount in May 2005 (NT dollars)
- PAY_AMT6: Previous payment amount in April 2005 (NT dollars) default.payment.next month: Default payment (1=yes, 0=no)

Complete the tasks below:

1. Import data "UCI_Credit_Card.csv"
2. Perform EDA and visualization (feel free to choose the data visualization that you think is most appropriate)
3. Prepare data for model training and separate it into training and test data
4. Train and evaluate the XG-Boost classification model
5. Train and evaluate the Support Vector Machine classification model
6. Train and evaluate a Naive Bayes classification model
7. Train and evaluate a Logistic Regression classification model
8. Train and evaluate the Random Forest classification model
9. Train and evaluate the K-Nearest Neighbors classification model
10. Plot the ROC curve for the entire model and calculate the AUC value
11. Which model performs best?

EDA

In [237…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [238…
```
credit_df = pd.read_csv('UCI_Credit_Card.csv')
```

In [239…
```
credit_df
```

Out[239]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	P
0	1	20000.0	2	2	1	24	2	2	-1	-1	...	0.0	0.0	0.0	
1	2	120000.0	2	2	2	26	-1	2	0	0	...	3272.0	3455.0	3261.0	
2	3	90000.0	2	2	2	34	0	0	0	0	...	14331.0	14948.0	15549.0	
3	4	50000.0	2	2	1	37	0	0	0	0	...	28314.0	28959.0	29547.0	
4	5	50000.0	1	2	1	57	-1	0	-1	0	...	20940.0	19146.0	19131.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000.0	1	3	1	39	0	0	0	0	...	88004.0	31237.0	15980.0	
29996	29997	150000.0	1	3	2	43	-1	-1	-1	-1	...	8979.0	5190.0	0.0	
29997	29998	30000.0	1	2	2	37	4	3	2	-1	...	20878.0	20582.0	19357.0	
29998	29999	80000.0	1	3	1	41	1	-1	0	0	...	52774.0	11855.0	48944.0	
29999	30000	50000.0	1	2	1	46	0	0	0	0	...	36535.0	32428.0	15313.0	

30000 rows × 25 columns

In [240…
```
credit_df.isna().sum()
```

Out[240]:
```
ID                          0
LIMIT_BAL                   0
SEX                         0
EDUCATION                   0
MARRIAGE                    0
AGE                         0
PAY_0                       0
PAY_2                       0
PAY_3                       0
PAY_4                       0
PAY_5                       0
PAY_6                       0
BILL_AMT1                   0
BILL_AMT2                   0
BILL_AMT3                   0
BILL_AMT4                   0
BILL_AMT5                   0
BILL_AMT6                   0
PAY_AMT1                    0
PAY_AMT2                    0
PAY_AMT3                    0
PAY_AMT4                    0
PAY_AMT5                    0
PAY_AMT6                    0
default.payment.next.month  0
dtype: int64
```

In [241…
```
credit_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          30000 non-null  int64
 1   LIMIT_BAL                   30000 non-null  float64
 2   SEX                         30000 non-null  int64
 3   EDUCATION                   30000 non-null  int64
 4   MARRIAGE                    30000 non-null  int64
 5   AGE                         30000 non-null  int64
 6   PAY_0                       30000 non-null  int64
 7   PAY_2                       30000 non-null  int64
 8   PAY_3                       30000 non-null  int64
 9   PAY_4                       30000 non-null  int64
 10  PAY_5                       30000 non-null  int64
 11  PAY_6                       30000 non-null  int64
 12  BILL_AMT1                   30000 non-null  float64
 13  BILL_AMT2                   30000 non-null  float64
 14  BILL_AMT3                   30000 non-null  float64
 15  BILL_AMT4                   30000 non-null  float64
 16  BILL_AMT5                   30000 non-null  float64
 17  BILL_AMT6                   30000 non-null  float64
 18  PAY_AMT1                    30000 non-null  float64
 19  PAY_AMT2                    30000 non-null  float64
 20  PAY_AMT3                    30000 non-null  float64
 21  PAY_AMT4                    30000 non-null  float64
 22  PAY_AMT5                    30000 non-null  float64
 23  PAY_AMT6                    30000 non-null  float64
 24  default.payment.next.month  30000 non-null  int64
dtypes: float64(13), int64(12)
memory usage: 5.7 MB
```

In [242…
```
credit_df.columns
```

Out[242]:
```
Index(['ID', 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0',
       'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6',
       'default.payment.next.month'],
      dtype='object')
```

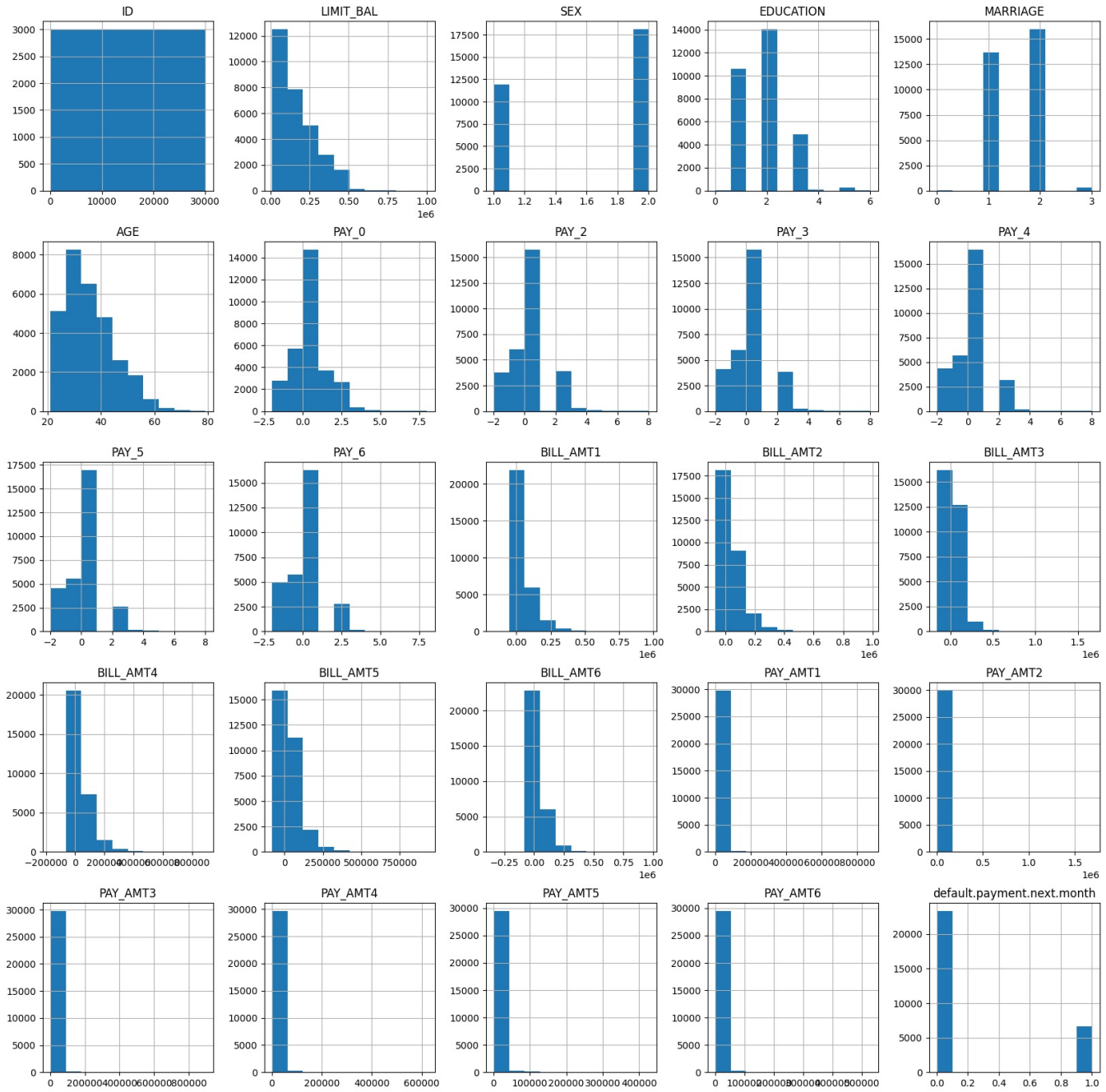
In [243…
```
credit_df.describe()
```

Out[243]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	3
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	35.485500	-0.016700	-0.133767	-0.166200	
std	8660.398374	129747.661567	0.489129	0.790349	0.521970	9.217904	1.123802	1.197186	1.196868	
min	1.000000	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.000000	-2.000000	-2.000000	
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.000000	-1.000000	-1.000000	
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000	0.000000	0.000000	
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	41.000000	0.000000	0.000000	0.000000	
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.000000	8.000000	8.000000	

8 rows × 25 columns

In [244…
```
credit_df.hist(figsize=(20,20))
plt.show()
```



In [245…
```
credit_df.drop(['ID'], axis = 1, inplace = True)
```

In [246…
```
credit_df
```

Out[246]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	I
0	20000.0	2	2	1	24	2	2	-1	-1	-2	...	0.0	0.0	0.0	
1	120000.0	2	2	2	26	-1	2	0	0	0	...	3272.0	3455.0	3261.0	
2	90000.0	2	2	2	34	0	0	0	0	0	...	14331.0	14948.0	15549.0	
3	50000.0	2	2	1	37	0	0	0	0	0	...	28314.0	28959.0	29547.0	
4	50000.0	1	2	1	57	-1	0	-1	0	0	...	20940.0	19146.0	19131.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	220000.0	1	3	1	39	0	0	0	0	0	...	88004.0	31237.0	15980.0	
29996	150000.0	1	3	2	43	-1	-1	-1	-1	0	...	8979.0	5190.0	0.0	
29997	30000.0	1	2	2	37	4	3	2	-1	0	...	20878.0	20582.0	19357.0	
29998	80000.0	1	3	1	41	1	-1	0	0	0	...	52774.0	11855.0	48944.0	
29999	50000.0	1	2	1	46	0	0	0	0	0	...	36535.0	32428.0	15313.0	

30000 rows × 24 columns

In [247…
```
default_df = credit_df[credit_df['default.payment.next.month']==1]
ndefault_df = credit_df[credit_df['default.payment.next.month']==0]
```

In [248…
```
print(len(default_df)/ len(credit_df))
```

```
0.2212
```

In [249…
```
print(len(ndefault_df)/ len(credit_df))
```

```
0.7788
```

In [250…
```
credit_df.corr()
```

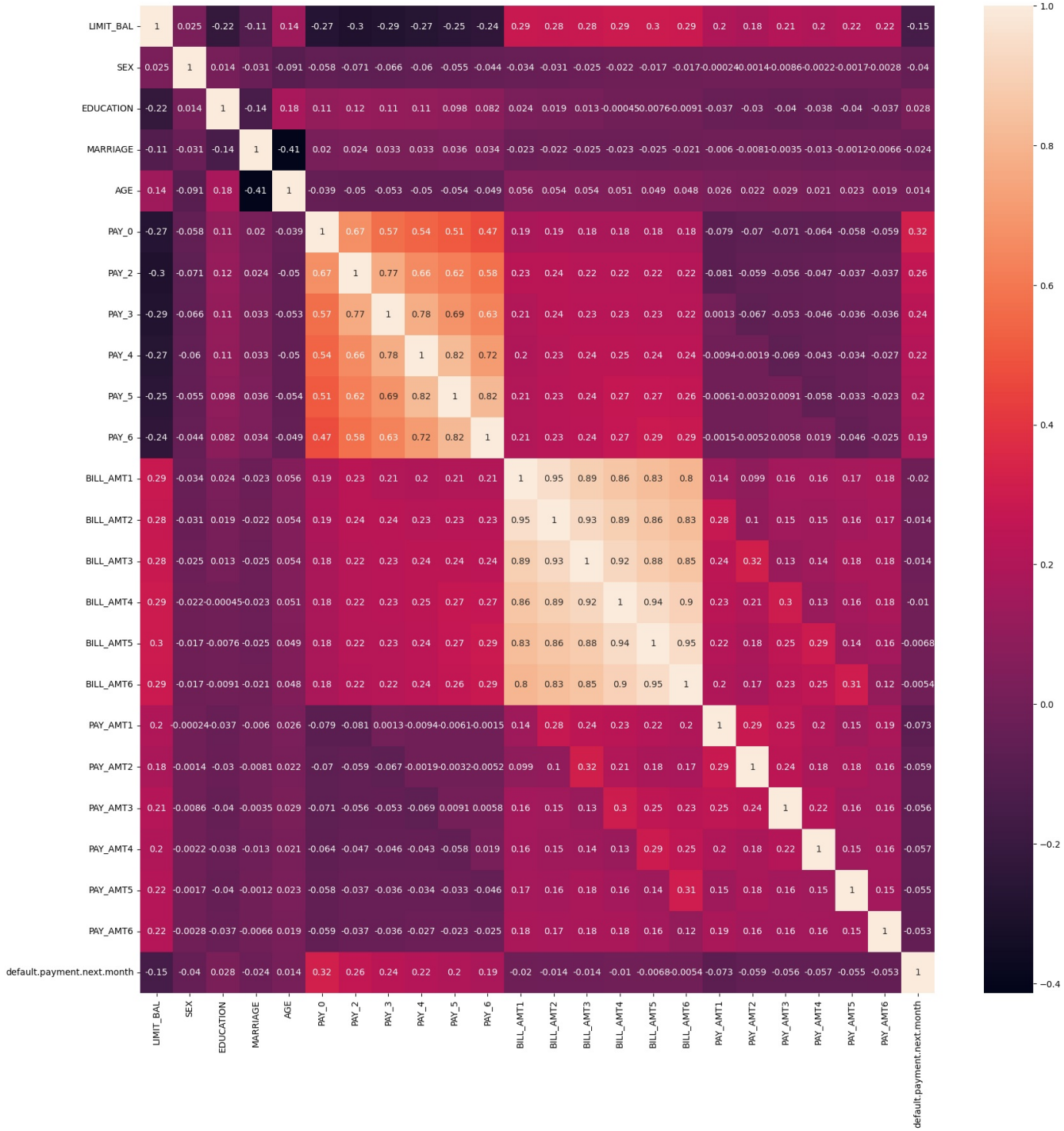
Out[250]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5
LIMIT_BAL	1.000000	0.024755	-0.219161	-0.108139	0.144713	-0.271214	-0.296382	-0.286123	-0.267460	-0.249411
SEX	0.024755	1.000000	0.014232	-0.031389	-0.090874	-0.057643	-0.070771	-0.066096	-0.060173	-0.055064
EDUCATION	-0.219161	0.014232	1.000000	-0.143464	0.175061	0.105364	0.121566	0.114025	0.108793	0.097520
MARRIAGE	-0.108139	-0.031389	-0.143464	1.000000	-0.414170	0.019917	0.024199	0.032688	0.033122	0.035629
AGE	0.144713	-0.090874	0.175061	-0.414170	1.000000	-0.039447	-0.050148	-0.053048	-0.049722	-0.053826
PAY_0	-0.271214	-0.057643	0.105364	0.019917	-0.039447	1.000000	0.672164	0.574245	0.538841	0.509426
PAY_2	-0.296382	-0.070771	0.121566	0.024199	-0.050148	0.672164	1.000000	0.766552	0.662067	0.622780
PAY_3	-0.286123	-0.066096	0.114025	0.032688	-0.053048	0.574245	0.766552	1.000000	0.777359	0.686775
PAY_4	-0.267460	-0.060173	0.108793	0.033122	-0.049722	0.538841	0.662067	0.777359	1.000000	0.819835
PAY_5	-0.249411	-0.055064	0.097520	0.035629	-0.053826	0.509426	0.622780	0.686775	0.819835	1.000000
PAY_6	-0.235195	-0.044008	0.082316	0.034345	-0.048773	0.474553	0.575501	0.632684	0.716449	0.816900
BILL_AMT1	0.285430	-0.033642	0.023581	-0.023472	0.056239	0.187068	0.234887	0.208473	0.202812	0.206684
BILL_AMT2	0.278314	-0.031183	0.018749	-0.021602	0.054283	0.189859	0.235257	0.237295	0.225816	0.226913
BILL_AMT3	0.283236	-0.024563	0.013002	-0.024909	0.053710	0.179785	0.224146	0.227494	0.244983	0.243335
BILL_AMT4	0.293988	-0.021880	-0.000451	-0.023344	0.051353	0.179125	0.222237	0.227202	0.245917	0.271915
BILL_AMT5	0.295562	-0.017005	-0.007567	-0.025393	0.049345	0.180635	0.221348	0.225145	0.242902	0.269783
BILL_AMT6	0.290389	-0.016733	-0.009099	-0.021207	0.047613	0.176980	0.219403	0.222327	0.239154	0.262509
PAY_AMT1	0.195236	-0.000242	-0.037456	-0.005979	0.026147	-0.079269	-0.080701	0.001295	-0.009362	-0.006089
PAY_AMT2	0.178408	-0.001391	-0.030038	-0.008093	0.021785	-0.070101	-0.058990	-0.066793	-0.001944	-0.003191
PAY_AMT3	0.210167	-0.008597	-0.039943	-0.003541	0.029247	-0.070561	-0.055901	-0.053311	-0.069235	0.009062
PAY_AMT4	0.203242	-0.002229	-0.038218	-0.012659	0.021379	-0.064005	-0.046858	-0.046067	-0.043461	-0.058299
PAY_AMT5	0.217202	-0.001667	-0.040358	-0.001205	0.022850	-0.058190	-0.037093	-0.035863	-0.033590	-0.033337
PAY_AMT6	0.219595	-0.002766	-0.037200	-0.006641	0.019478	-0.058673	-0.036500	-0.035861	-0.026565	-0.023027
default.payment.next.month	-0.153520	-0.039961	0.028006	-0.024339	0.013890	0.324794	0.263551	0.235253	0.216614	0.204149

24 rows × 24 columns

In [251…
```
plt.figure(figsize=(20,20))
sns.heatmap(credit_df.corr(), annot = True)
```

Out[251]: <Axes: >



In [252…

```
credit_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 24 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   LIMIT_BAL                   30000 non-null  float64
 1   SEX                         30000 non-null  int64
 2   EDUCATION                   30000 non-null  int64
 3   MARRIAGE                    30000 non-null  int64
 4   AGE                         30000 non-null  int64
 5   PAY_0                       30000 non-null  int64
 6   PAY_2                       30000 non-null  int64
 7   PAY_3                       30000 non-null  int64
 8   PAY_4                       30000 non-null  int64
 9   PAY_5                       30000 non-null  int64
 10  PAY_6                       30000 non-null  int64
 11  BILL_AMT1                   30000 non-null  float64
 12  BILL_AMT2                   30000 non-null  float64
 13  BILL_AMT3                   30000 non-null  float64
 14  BILL_AMT4                   30000 non-null  float64
 15  BILL_AMT5                   30000 non-null  float64
 16  BILL_AMT6                   30000 non-null  float64
 17  PAY_AMT1                    30000 non-null  float64
 18  PAY_AMT2                    30000 non-null  float64
 19  PAY_AMT3                    30000 non-null  float64
 20  PAY_AMT4                    30000 non-null  float64
 21  PAY_AMT5                    30000 non-null  float64
 22  PAY_AMT6                    30000 non-null  float64
 23  default.payment.next.month  30000 non-null  int64
dtypes: float64(13), int64(11)
memory usage: 5.5 MB
```

In [253…
```
categorical_column = credit_df[['SEX', 'EDUCATION','MARRIAGE']]
categorical_column
```

Out[253]:

	SEX	EDUCATION	MARRIAGE
0	2	2	1
1	2	2	2
2	2	2	2
3	2	2	1
4	1	2	1
...	...	...	...
29995	1	3	1
29996	1	3	2
29997	1	2	2
29998	1	3	1
29999	1	2	1

30000 rows × 3 columns

In [254…
```
numerical_column = credit_df[['LIMIT_BAL', 'AGE', 'PAY_0',
       'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6']]
```

In [255…
```
numerical_column
```

Out[255]:

	LIMIT_BAL	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BIL
0	20000.0	24	2	2	-1	-1	-2	-2	3913.0	3102.0	689.0	0.0	0.0	
1	120000.0	26	-1	2	0	0	0	2	2682.0	1725.0	2682.0	3272.0	3455.0	
2	90000.0	34	0	0	0	0	0	0	29239.0	14027.0	13559.0	14331.0	14948.0	
3	50000.0	37	0	0	0	0	0	0	46990.0	48233.0	49291.0	28314.0	28959.0	
4	50000.0	57	-1	0	-1	0	0	0	8617.0	5670.0	35835.0	20940.0	19146.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	220000.0	39	0	0	0	0	0	0	188948.0	192815.0	208365.0	88004.0	31237.0	
29996	150000.0	43	-1	-1	-1	-1	0	0	1683.0	1828.0	3502.0	8979.0	5190.0	
29997	30000.0	37	4	3	2	-1	0	0	3565.0	3356.0	2758.0	20878.0	20582.0	
29998	80000.0	41	1	-1	0	0	0	-1	-1645.0	78379.0	76304.0	52774.0	11855.0	
29999	50000.0	46	0	0	0	0	0	0	47929.0	48905.0	49764.0	36535.0	32428.0	

30000 rows × 20 columns

In [256…
```
from sklearn.preprocessing import OneHotEncoder
```

In [257…
```
encoder = OneHotEncoder()
one_hot_encoded = encoder.fit_transform(categorical_column)
```

In [258…
```
categorical_column = one_hot_encoded.toarray()
```

In [259…
```
categorical_column = pd.DataFrame(categorical_column)
```

In [260…
```
categorical_column
```

Out[260]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
2	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
3	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
4	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
29996	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
29997	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
29998	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
29999	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0

30000 rows × 13 columns

In [261…
```
credit_df2 = pd.concat([categorical_column, numerical_column], axis = 1)
```

In [262…
```
credit_df2
```

Out[262]:

	0	1	2	3	4	5	6	7	8	9	...	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AM
0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	689.0	0.0	0.0	0.0	0.0	689.0	
1	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	2682.0	3272.0	3455.0	3261.0	0.0	1000.0	100
2	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	13559.0	14331.0	14948.0	15549.0	1518.0	1500.0	100
3	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	49291.0	28314.0	28959.0	29547.0	2000.0	2019.0	120
4	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	35835.0	20940.0	19146.0	19131.0	2000.0	36681.0	1000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	208365.0	88004.0	31237.0	15980.0	8500.0	20000.0	500
29996	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	3502.0	8979.0	5190.0	0.0	1837.0	3526.0	899
29997	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	2758.0	20878.0	20582.0	19357.0	0.0	0.0	2200
29998	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	76304.0	52774.0	11855.0	48944.0	85900.0	3409.0	117
29999	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	49764.0	36535.0	32428.0	15313.0	2078.0	1800.0	143

30000 rows × 33 columns

In [263…
```
credit_df2.columns = credit_df2.columns.astype(str)
```

In [264…
```
from sklearn.preprocessing import MinMaxScaler
```

In [265…
```
scalar = MinMaxScaler()
```

In [266…
```
X = scalar.fit_transform(credit_df2)
```

In [267…
```
y = credit_df['default.payment.next.month']
```

In [268…
```
from sklearn.model_selection import train_test_split
```

In [269…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0, test_size = 0.2)
```

Latih dan evaluasi model klasifikasi XG-Boost

In [277…
```
import xgboost as xgb
```

In [279…
```
xgb_model = xgb.XGBClassifier()
xgb_model.fit(X_train, y_train)
```

Out[279]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=Non
e,
              enable_categorical=False, eval_metric=None, feature_types=Non
e,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=Non
e,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
```

In [280…
```
y_predict = xgb_model.predict(X_test)
```

In [281…
```
y_predict
```

Out[281]:
```
array([1, 0, 0, ..., 0, 1, 0])
```

In [288…
```
y_test
```

Out[288]:
```
8225     0
10794    0
9163     0
26591    0
6631     0
        ..
12715    0
28867    0
3758     1
17842    0
9119     0
Name: default.payment.next.month, Length: 6000, dtype: int64
```

In [287…
```
from sklearn.metrics import classification_report
```

In [289…
```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.85      0.94      0.89      4703
           1       0.65      0.38      0.48      1297

    accuracy                           0.82      6000
   macro avg       0.75      0.66      0.68      6000
weighted avg       0.80      0.82      0.80      6000
```

Latih dan evaluasi model klasifikasi Support Vector Machine

In [337…
```
from sklearn.svm import SVC
```

In [338…
```
svm_model = SVC(probability=True)
svm_model.fit(X_train, y_train)
```

Out[338]:
```
SVC(probability=True)
```

In [342…
```
y_predict = svm_model.predict(X_test)
```

In [343…
```
y_test
```

Out[343]:
```
8225     0
10794    0
9163     0
26591    0
6631     0
        ..
12715    0
28867    0
3758     1
17842    0
9119     0
Name: default.payment.next.month, Length: 6000, dtype: int64
```

In [344…
```
print(classification_report(y_test, y_predict))
```

```
                precision    recall  f1-score   support

           0       0.83      0.97      0.89      4703
           1       0.69      0.26      0.38      1297

    accuracy                           0.81      6000
   macro avg       0.76      0.61      0.63      6000
weighted avg       0.80      0.81      0.78      6000
```

Latih dan evaluasi model klasifikasi Naive Bayes

In [299…
```
from sklearn.naive_bayes import GaussianNB
```

In [300…
```
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
```

Out[300]:
```
▾ GaussianNB
GaussianNB()
```

In [301…
```
y_predict = nb_model.predict(X_test)
```

In [302…
```
y_test
```

Out[302]:
```
8225     0
10794    0
9163     0
26591    0
6631     0
        ..
12715    0
28867    0
3758     1
17842    0
9119     0
Name: default.payment.next.month, Length: 6000, dtype: int64
```

In [303…
```
print(classification_report(y_test, y_predict))
```

```
                precision    recall  f1-score   support

           0       0.91      0.10      0.18      4703
           1       0.23      0.97      0.37      1297

    accuracy                           0.28      6000
   macro avg       0.57      0.53      0.27      6000
weighted avg       0.76      0.28      0.22      6000
```

Latih dan evaluasi model klasifikasi Logistic Regression

In [331…
```
from sklearn.linear_model import LogisticRegression
```

In [332…
```
lr_model = LogisticRegression(max_iter = 10000)
lr_model.fit(X_train,y_train)
```

Out[332]:
```
▾ LogisticRegression
LogisticRegression(max_iter=10000)
```

In [333…
```
y_predict = lr_model.predict(X_test)
```

In [334…
```
y_test
```

Out[334]:
```
8225     0
10794    0
9163     0
26591    0
6631     0
        ..
12715    0
28867    0
3758     1
17842    0
9119     0
Name: default.payment.next.month, Length: 6000, dtype: int64
```

In [335…
```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.82      0.98      0.89      4703
           1       0.77      0.23      0.35      1297

    accuracy                           0.82      6000
   macro avg       0.79      0.60      0.62      6000
weighted avg       0.81      0.82      0.78      6000
```

Latih dan evaluasi model klasifikasi Random Forest

In [316…
```
from sklearn.ensemble import RandomForestClassifier
```

In [318…
```
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
```

Out[318]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [319…
```
y_predict = rf_model.predict(X_test)
```

In [320…
```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      4703
           1       0.66      0.37      0.47      1297

    accuracy                           0.82      6000
   macro avg       0.75      0.66      0.68      6000
weighted avg       0.80      0.82      0.80      6000
```

Latih dan evaluasi model klasifikasi K-Nearest Neighbors

In [322…
```
from sklearn.neighbors import KNeighborsClassifier
```

In [323…
```
knn_model = KNeighborsClassifier()
knn_model.fit(X_train, y_train)
```

Out[323]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

In [324…
```
y_predict = knn_model.predict(X_test)
```

In [325…
```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.83      0.92      0.87      4703
           1       0.53      0.34      0.41      1297

    accuracy                           0.79      6000
   macro avg       0.68      0.63      0.64      6000
weighted avg       0.77      0.79      0.77      6000
```

In [345…
```
from sklearn.metrics import roc_curve

fpr1, tpr1, thresh1 = roc_curve(y_test, xgb_model.predict_proba(X_test)[:, 1], pos_label = 1)
fpr2, tpr2, thresh2 = roc_curve(y_test, svm_model.predict_proba(X_test)[:, 1], pos_label = 1)
fpr3, tpr3, thresh3 = roc_curve(y_test, nb_model.predict_proba(X_test)[:, 1], pos_label = 1)
fpr4, tpr4, thresh4 = roc_curve(y_test, lr_model.predict_proba(X_test)[:, 1], pos_label = 1)
fpr5, tpr5, thresh5 = roc_curve(y_test, rf_model.predict_proba(X_test)[:, 1], pos_label = 1)
fpr6, tpr6, thresh6 = roc_curve(y_test, knn_model.predict_proba(X_test)[:, 1], pos_label = 1)
```

In [347…
```
from sklearn.metrics import roc_auc_score

auc_score1 = roc_auc_score(y_test, xgb_model.predict_proba(X_test)[:, 1])
auc_score2 = roc_auc_score(y_test, svm_model.predict_proba(X_test)[:, 1])
auc_score3 = roc_auc_score(y_test, nb_model.predict_proba(X_test)[:, 1])
auc_score4 = roc_auc_score(y_test, lr_model.predict_proba(X_test)[:, 1])
auc_score5 = roc_auc_score(y_test, rf_model.predict_proba(X_test)[:, 1])
auc_score6 = roc_auc_score(y_test, knn_model.predict_proba(X_test)[:, 1])
```

In [348…
```
print("XGB: ", auc_score1)
print("SVM: ", auc_score2)
print("NAIVE_BAYES: ", auc_score3)
print("LINEAR_REGRESSION: ", auc_score4)
print("RANDOM_FOREST: ", auc_score5)
print("KNN: ", auc_score6)
```

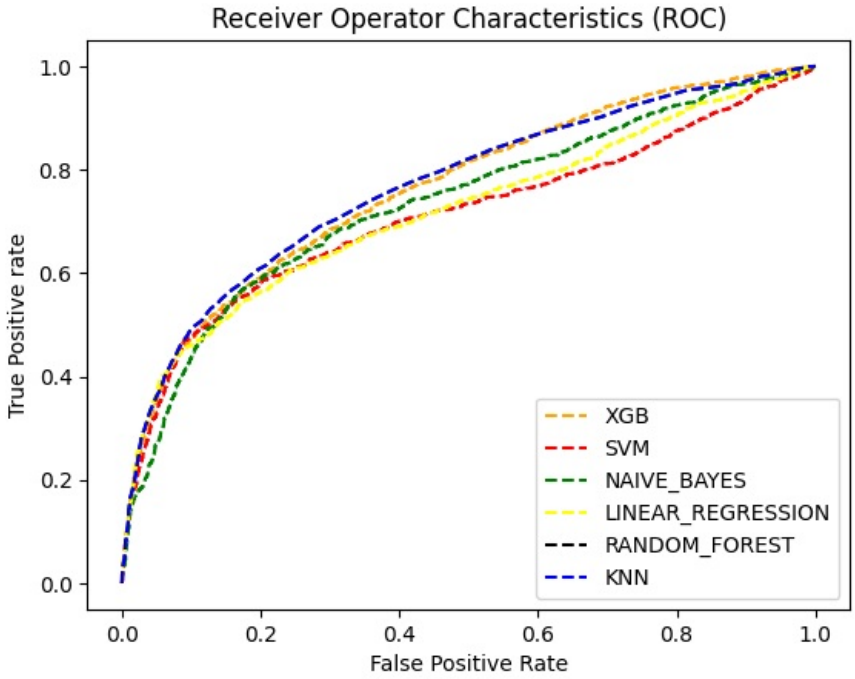
```
XGB:  0.7637615780606254
SVM:  0.7094111093314509
NAIVE_BAYES:  0.7341428583372775
LINEAR_REGRESSION:  0.7189789781322016
RANDOM_FOREST:  0.7678342585836136
KNN:  0.6973296298184644
```

In [349…

```
plt.plot(fpr1, tpr1, linestyle = "--", color = "orange", label = "XGB")
plt.plot(fpr2, tpr2, linestyle = "--", color = "red", label = "SVM")
plt.plot(fpr3, tpr3, linestyle = "--", color = "green", label = "NAIVE_BAYES")
plt.plot(fpr4, tpr4, linestyle = "--", color = "yellow", label = "LINEAR_REGRESSION")
plt.plot(fpr5, tpr5, linestyle = "--", color = "black", label = "RANDOM_FOREST")
plt.plot(fpr5, tpr5, linestyle = "--", color = "blue", label = "KNN")


plt.title('Receiver Operator Characteristics (ROC)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')

plt.legend(loc = 'best')
plt.savefig('ROC', dpi = 300)
plt.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js