Komal Kumari

Project Name: Prediction for Credit Card Approval

```
Welcome to the Machine Learning Internship program, focused on Credit Card Approval Prediction. In
this project, your task is to utilize machine learning techniques to predict whether an applicant will be
approved for a credit card or not.
```

Problem Statement:

```
The primary objective of this project is to predict the approval or rejection of credit card applications.
The challenge lies in understanding the key factors influencing credit card approval decisions and
building a predictive model to assist in the decision-making process.
```

Your Mission:

```
1. Exploratory Data Analysis (EDA):
2. Feature Engineering:
3. Data Preprocessing:
4. Machine Learning Model Development:
5. Model Evaluation:
6. ML Pipelines and Model Deployment:
7. Recommendations:
```

In [9]:
```
#IMPORT LIABRARIES
```

In [10]:
```
# Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
```

In [ ]:

In [4]:
```
df = pd.read_csv('train_data.csv')
df
```

Out[4]:

	ID	Gender	Has a car	Has a property	Children count	Income	Employment status	Education level	Marital status	Dwelling	Age	Employment length	Has a mobile phone	Has a work phone	Has a phone	Has an email
0	5037048	M	Y	Y	0	135000.0	Working	Secondary / secondary special	Married	With parents	-16271	-3111	1	0	0	0
1	5044630	F	Y	N	1	135000.0	Commercial associate	Higher education	Single / not married	House / apartment	-10130	-1651	1	0	0	0
2	5079079	F	N	Y	2	180000.0	Commercial associate	Secondary / secondary special	Married	House / apartment	-12821	-5657	1	0	0	0
3	5112872	F	Y	Y	0	360000.0	Commercial associate	Higher education	Single / not married	House / apartment	-20929	-2046	1	0	0	1
4	5105858	F	N	N	0	270000.0	Working	Secondary / secondary special	Separated	House / apartment	-16207	-515	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

In [11]:
```
df.head()
```

Out[11]:

	ID	Gender	Has a car	Has a property	Children count	Income	Employment status	Education level	Marital status	Dwelling	Age	Employment length	Has a mobile phone	Has a work phone	Has a phone	Has an email	J
0	5037048	M	Y	Y	0	135000.0	Working	Secondary / secondary special	Married	With parents	-16271	-3111	1	0	0	0	C
1	5044630	F	Y	N	1	135000.0	Commercial associate	Higher education	Single / not married	House / apartment	-10130	-1651	1	0	0	0	Acco
2	5079079	F	N	Y	2	180000.0	Commercial associate	Secondary / secondary special	Married	House / apartment	-12821	-5657	1	0	0	0	L
3	5112872	F	Y	Y	0	360000.0	Commercial associate	Higher education	Single / not married	House / apartment	-20929	-2046	1	0	0	1	M
4	5105858	F	N	N	0	270000.0	Working	Secondary / secondary special	Separated	House / apartment	-16207	-515	1	0	1	0	

In [13]:
```
df.shape
```
Out[13]: (29165, 20)

In [14]:
```
df.tail()
```
Out[14]:

	ID	Gender	Has a car	Has a property	Children count	Income	Employment status	Education level	Marital status	Dwelling	Age	Employment length	Has a mobile phone	Has a work phone	Has a phone	Has an email	
29160	5067139	F	N	Y	0	112500.0	Pensioner	Secondary / secondary special	Single / not married	House / apartment	-23400	365243	1	0	1	1	
29161	5029193	F	N	Y	1	135000.0	Commercial associate	Secondary / secondary special	Married	House / apartment	-15532	-8256	1	0	0	0	C
29162	5047710	F	N	Y	0	76500.0	Working	Secondary / secondary special	Married	House / apartment	-17782	-3291	1	1	1	0	M
29163	5009886	F	N	Y	0	157500.0	Pensioner	Secondary / secondary special	Civil marriage	House / apartment	-21635	365243	1	0	1	0	
29164	5062632	F	N	Y	0	585000.0	Commercial associate	Secondary / secondary	Married	House / apartment	-18858	-2010	1	0	1	0	

In [11]:
```
df.head()
```

Out[11]:

	ID	Gender	Has a car	Has a property	Children count	Income	Employment status	Education level	Marital status	Dwelling	Age	Employment length	Has a mobile phone	Has a work phone	Has a phone	Has an email	J
0	5037048	M	Y	Y	0	135000.0	Working	Secondary / secondary special	Married	With parents	-16271	-3111	1	0	0	0	Ci
1	5044630	F	Y	N	1	135000.0	Commercial associate	Higher education	Single / not married	House / apartment	-10130	-1651	1	0	0	0	Acco
2	5079079	F	N	Y	2	180000.0	Commercial associate	Secondary / secondary special	Married	House / apartment	-12821	-5657	1	0	0	0	L
3	5112872	F	Y	Y	0	360000.0	Commercial associate	Higher education	Single / not married	House / apartment	-20929	-2046	1	0	0	1	Ma
4	5105858	F	N	N	0	270000.0	Working	Secondary / secondary special	Separated	House / apartment	-16207	-515	1	0	1	0	

In [13]:
```
df.shape
```

Out[13]: (29165, 20)

In [14]:
```
df.tail()
```

Out[14]:

	ID	Gender	Has a car	Has a property	Children count	Income	Employment status	Education level	Marital status	Dwelling	Age	Employment length	Has a mobile phone	Has a work phone	Has a phone	Has an email	
29160	5067139	F	N	Y	0	112500.0	Pensioner	Secondary / secondary special	Single / not married	House / apartment	-23400	365243	1	0	1	1	
29161	5029193	F	N	Y	1	135000.0	Commercial associate	Secondary / secondary special	Married	House / apartment	-15532	-8256	1	0	0	0	(
								Secondary									

In [15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29165 entries, 0 to 29164
Data columns (total 20 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   29165 non-null  int64  
 1   Gender               29165 non-null  object 
 2   Has a car            29165 non-null  object 
 3   Has a property       29165 non-null  object 
 4   Children count       29165 non-null  int64  
 5   Income               29165 non-null  float64
 6   Employment status    29165 non-null  object 
 7   Education level      29165 non-null  object 
 8   Marital status       29165 non-null  object 
 9   Dwelling             29165 non-null  object 
 10  Age                  29165 non-null  int64  
 11  Employment length    29165 non-null  int64  
 12  Has a mobile phone   29165 non-null  int64  
 13  Has a work phone     29165 non-null  int64  
 14  Has a phone          29165 non-null  int64  
 15  Has an email         29165 non-null  int64  
 16  Job title            20138 non-null  object 
 17  Family member count  29165 non-null  float64
 18  Account age          29165 non-null  float64
 19  Is high risk         29165 non-null  int64  
dtypes: float64(3), int64(9), object(8)
memory usage: 4.5+ MB
```

In [17]:
```
#insight
# ID,Children count, Income, Age, Employment length,Has a mobile phone,Has a phone,Has an email,Family member count,Account age,
# other object
```

In [18]:
```
#***Exploratory Data Analysis (EDA)
```

In [19]:
```
df.describe()
```

Out[19]:

	ID	Children count	Income	Age	Employment length	Has a mobile phone	Has a work phone	Has a phone	Has an email	Family member count	Account age
count	2.916500e+04	29165.000000	2.916500e+04	29165.000000	29165.000000	29165.0	29165.000000	29165.000000	29165.000000	29165.000000	29165.00000

In [19]:
```python
df.describe()
```

Out[19]:

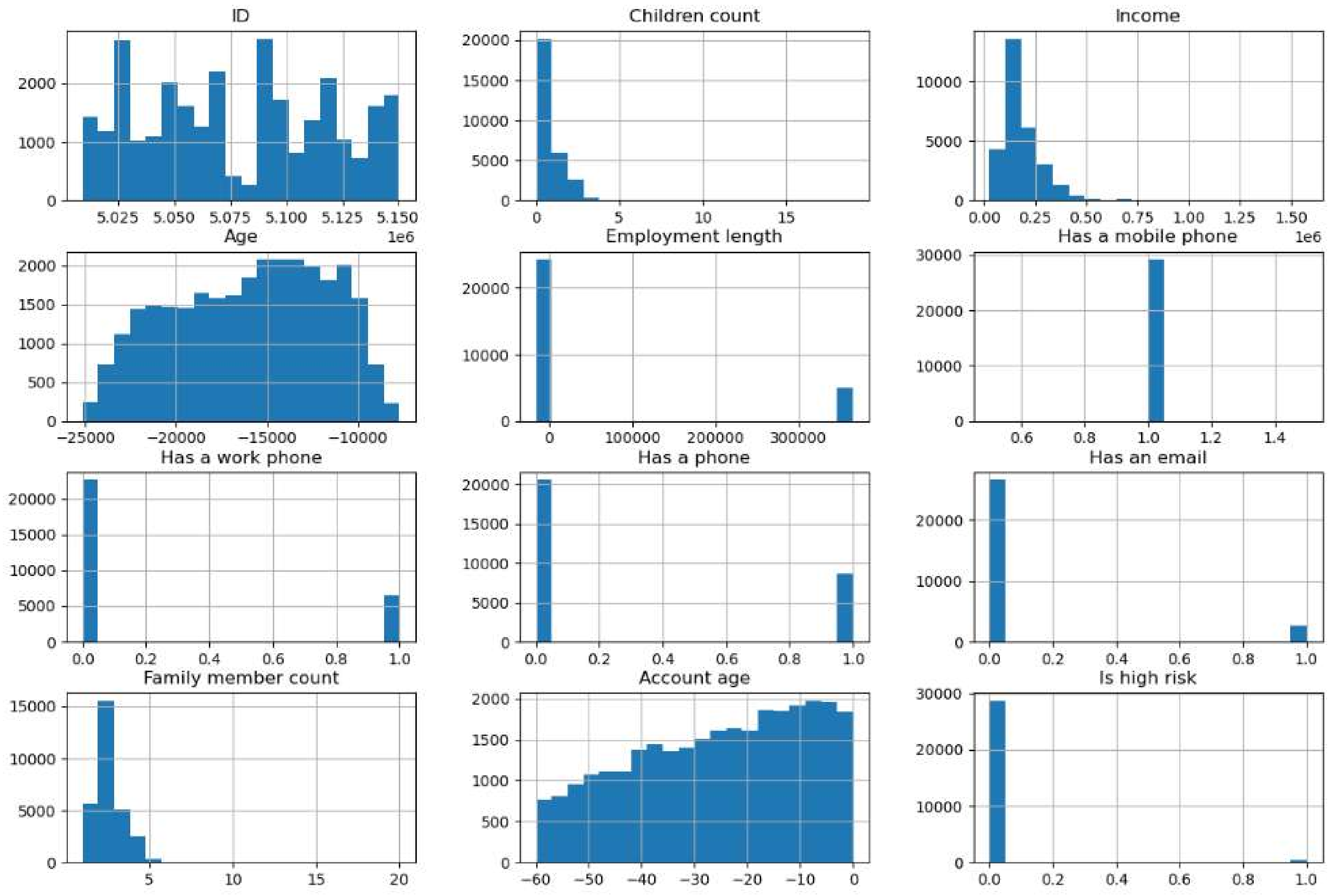
	ID	Children count	Income	Age	Employment length	Has a mobile phone	Has a work phone	Has a phone	Has an email	Family member count	Account age
count	2.916500e+04	29165.000000	2.916500e+04	29165.000000	29165.000000	29165.0	29165.000000	29165.000000	29165.000000	29165.000000	29165.00000
mean	5.078232e+06	0.430790	1.868904e+05	-15979.477490	59257.761255	1.0	0.224310	0.294977	0.090279	2.197531	-26.13773
std	4.182400e+04	0.741882	1.014096e+05	4202.997485	137655.883458	0.0	0.417134	0.456040	0.286587	0.912189	16.48670
min	5.008804e+06	0.000000	2.700000e+04	-25152.000000	-15713.000000	1.0	0.000000	0.000000	0.000000	1.000000	-60.00000
25%	5.042047e+06	0.000000	1.215000e+05	-19444.000000	-3153.000000	1.0	0.000000	0.000000	0.000000	2.000000	-39.00000
50%	5.074666e+06	0.000000	1.575000e+05	-15565.000000	-1557.000000	1.0	0.000000	0.000000	0.000000	2.000000	-24.00000
75%	5.114629e+06	1.000000	2.250000e+05	-12475.000000	-412.000000	1.0	0.000000	1.000000	0.000000	3.000000	-12.00000
max	5.150485e+06	19.000000	1.575000e+06	-7705.000000	365243.000000	1.0	1.000000	1.000000	1.000000	20.000000	0.00000

In [20]:
```python
# Checking for missing values
print(df.isnull().sum())
```

```
ID                        0
Gender                    0
Has a car                 0
Has a property            0
Children count            0
Income                    0
Employment status         0
Education level           0
Marital status            0
Dwelling                  0
Age                       0
Employment length         0
Has a mobile phone        0
Has a work phone          0
Has a phone               0
Has an email              0
Job title              9027
Family member count       0
Account age               0
Is high risk              0
dtype: int64
```

In [21]:

```
# Visualize the distribution of numerical features
df.hist(bins=20, figsize=(15,10))
plt.show()
```



In [68]:
```
# Count plot for categorical features
plt.figure(figsize =(5,5))
for col in ['Gender', 'Has a car', 'Has a property', 'Employment status', 'Education level', 'Marital status', 'Dwelling']:
    sns.countplot(y=col, data=df)
    plt.show()
```

4. Feature Engineering

In [23]:
```
# Creating interaction terms (example)
df['Income_per_family_member'] = df['Income'] / (df['Family member count'] + 1)  # Avoid division by zero

# Creating binary features (example)
df['Is_Employed'] = df['Employment status'].apply(lambda x: 1 if x != 'Unemployed' else 0)
```

In [ ]:

5. Data Preprocessing

In [81]:
```
df.isnull().sum()
```

5. Data Preprocessing

In [81]:
```
df.isnull().sum()
```

Out[81]:
```
ID                           0
Gender                       0
Has a car                    0
Has a property               0
Children count               0
Income                       0
Employment status            0
Education level              0
Marital status               0
Dwelling                     0
Age                          0
Employment length            0
Has a mobile phone           0
Has a work phone             0
Has a phone                  0
Has an email                 0
Job title                 9027
Family member count          0
Account age                  0
Is high risk                 0
Income_per_family_member     0
Is_Employed                  0
dtype: int64
```

In [27]:
```
duplicates = df['Job title'].duplicated(keep=False)
print(df[duplicates])  # Print rows where 'Job title' is duplicated
```

```
            ID Gender Has a car Has a property  Children count    Income  \
0      5037048      M         Y              Y               0  135000.0
1      5044630      F         Y              N               1  135000.0
2      5079079      F         N              Y               2  180000.0
3      5112872      F         Y              Y               0  360000.0
4      5105858      F         N              N               0  270000.0
...        ...    ...       ...            ...             ...       ...
29160  5067139      F         N              Y               0  112500.0
29161  5029193      F         N              Y               1  135000.0
29162  5047710      F         N              Y               0   76500.0
29163  5009886      F         N              Y               0  157500.0
29164  5062632      F         N              Y               0  585000.0
```

```
29160  Single / not married  House / apartment  ...                     1
29161               Married  House / apartment  ...                     1
29162               Married  House / apartment  ...                     1
29163        Civil marriage  House / apartment  ...                     1
29164               Married  House / apartment  ...                     1

       Has a work phone  Has a phone  Has an email    Job title  \
0                     0            0             0   Core staff
1                     0            0             0  Accountants
2                     0            0             0     Laborers
3                     0            0             1     Managers
4                     0            1             0          NaN
...                 ...          ...           ...          ...
29160                 0            1             1          NaN
29161                 0            0             0   Core staff
29162                 1            1             0     Managers
29163                 0            1             0          NaN
29164                 0            1             0          NaN

       Family member count Account age  Is high risk  \
0                      2.0       -17.0             0
1                      2.0        -1.0             0
2                      4.0       -38.0             0
3                      1.0       -11.0             0
4                      1.0       -41.0             0
...                    ...         ...           ...
29160                  1.0        -5.0             0
29161                  3.0       -24.0             0
29162                  2.0       -29.0             0
29163                  2.0       -37.0             0
29164                  2.0       -43.0             0

       Income_per_family_member  Is_Employed
0                       45000.0            1
1                       45000.0            1
2                       36000.0            1
3                      180000.0            1
4                      135000.0            1
...                         ...          ...
29160                   56250.0            1
29161                   33750.0            1
29162                   25500.0            1
29163                   52500.0            1
29164                  195000.0            1

[29165 rows x 22 columns]
```

In [28]:
```python
mode_job_title = df['Job title'].mode()[0]
```

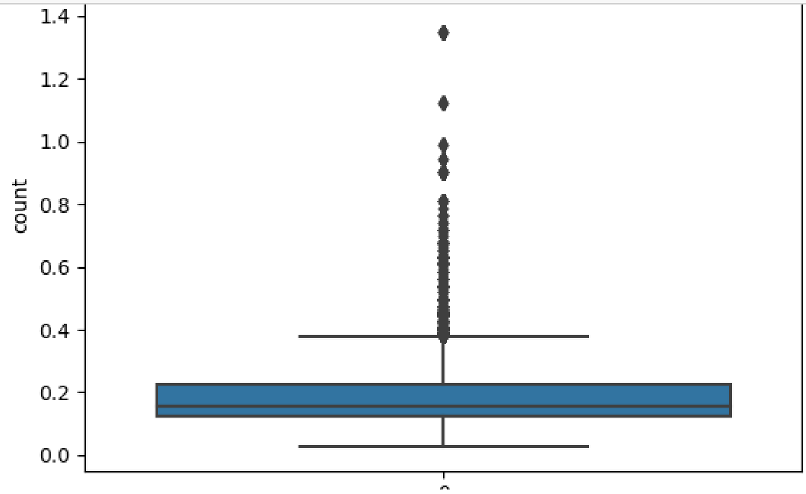
In [29]:
```python
df.loc[duplicates, 'Job title'] = mode_job_title
```

In [30]:
```python
df.isnull().sum()
```

Out[30]:
```
ID                          0
Gender                      0
Has a car                   0
Has a property              0
Children count              0
Income                      0
Employment status           0
Education level             0
Marital status              0
Dwelling                    0
Age                         0
Employment length           0
Has a mobile phone          0
Has a work phone            0
Has a phone                 0
Has an email                0
Job title                   0
Family member count         0
Account age                 0
Is high risk                0
Income_per_family_member    0
Is_Employed                 0
dtype: int64
```

In [37]:
```python
# Grabbing all the columns from the dataset
col_list = df.columns

# we need only the numerical data
for x in col_list:
  if df[x].dtype != "object":
    sns.boxplot(df[x])
    plt.xlabel(x)
    plt.ylabel("count")
    plt.show()
```

In [37]:
```
# Grabbing all the columns from the dataset
col_list = df.columns

# we need only the numerical data
for x in col_list:
  if df[x].dtype != "object":
    sns.boxplot(df[x])
    plt.xlabel(x)
    plt.ylabel("count")
    plt.show()
```



In [38]:
```
# Label Encoding for categorical variables
# Label Encoding for categorical variables
from sklearn.preprocessing import LabelEncoder

# List of columns to be label encoded
label_encode_cols = ['Gender', 'Has a car', 'Has a property', 'Employment status',
                     'Education level', 'Marital status', 'Dwelling', 'Job title']

label_encoders = {}  # Dictionary to store label encoders for each column

for col in label_encode_cols:
    label_encoders[col] = LabelEncoder()
    df[col] = label_encoders[col].fit_transform(df[col])

df.head()  # Check the first few rows to ensure the encoding worked
```

t[38]:

	ID	Gender	Has a car	Has a property	Children count	Income	Employment status	Education level	Marital status	Dwelling	...	Has a mobile phone	Has a work phone	Has a phone	Has an email	Job title	Family member count	Account age
0	5037048	1	1	1	0	135000.0	4	4	1	5	...	1	0	0	0	0	2.0	-17.0
1	5044630	0	1	0	1	135000.0	0	1	3	1	...	1	0	0	0	0	2.0	-1.0
2	5079079	0	0	1	2	180000.0	0	4	1	1	...	1	0	0	0	0	4.0	-38.0
3	5112872	0	1	1	0	360000.0	0	1	3	1	...	1	0	0	1	0	1.0	-11.0
4	5105858	0	0	0	0	270000.0	4	4	2	1	...	1	0	1	0	0	1.0	-41.0

5 rows × 22 columns

[39]:

```python
# Ensure binary columns are integers (0 and 1)
binary_cols = ['Has a mobile phone', 'Has a work phone', 'Has a phone', 'Has an email']

for col in binary_cols:
    df[col] = df[col].astype(int)
```

6. Split the Dataset

[40]:

```python
# Split the dataset into features and target variable
X = df.drop(['ID', 'Is high risk'], axis=1)  # Drop target variable and any ID columns
y = df['Is high risk']

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

7. Train Machine Learning Models

[41]:

```python
# Initialize models
log_reg = LogisticRegression()
dec_tree = DecisionTreeClassifier()
```

7. Train Machine Learning Models

In [41]:
```
# Initialize models
log_reg = LogisticRegression()
dec_tree = DecisionTreeClassifier()
rand_forest = RandomForestClassifier()
grad_boost = GradientBoostingClassifier()
```

In [42]:
```
# Train the models
log_reg.fit(X_train, y_train)
dec_tree.fit(X_train, y_train)
rand_forest.fit(X_train, y_train)
grad_boost.fit(X_train, y_train)
```

Out[42]:
```
GradientBoostingClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

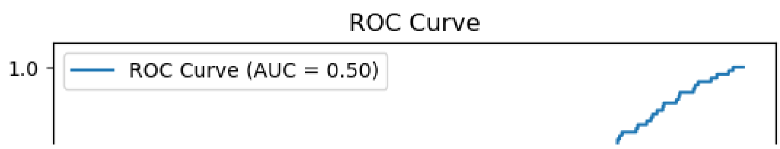
In [43]:
```
# Function to evaluate models
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
    print(f"ROC AUC Score: {roc_auc_score(y_test, y_pred):.2f}")
    fpr, tpr, thresholds = roc_curve(y_test, model.predict_proba(X_test)[:,1])
    plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc_score(y_test, y_pred):.2f})')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')
    plt.legend()
    plt.show()
```

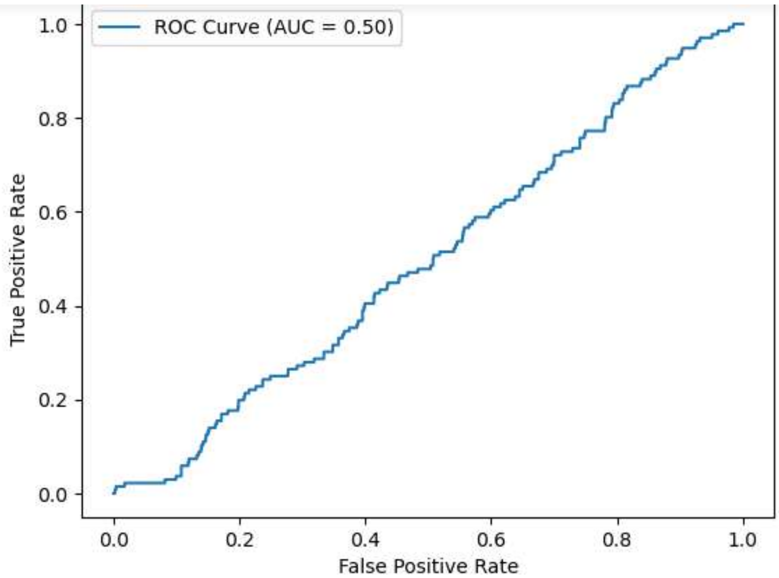
In [44]:
```
# Evaluate each model
print("Logistic Regression")
evaluate_model(log_reg, X_test, y_test)
```

In [44]:
```python
# Evaluate each model
print("Logistic Regression")
evaluate_model(log_reg, X_test, y_test)
```

```
Logistic Regression
Accuracy: 0.98
Confusion Matrix:
[[8614    0]
 [ 136    0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      8614
           1       0.00      0.00      0.00       136

    accuracy                           0.98      8750
   macro avg       0.49      0.50      0.50      8750
weighted avg       0.97      0.98      0.98      8750

ROC AUC Score: 0.50
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-sco
re are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-sco
re are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-sco
re are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
```

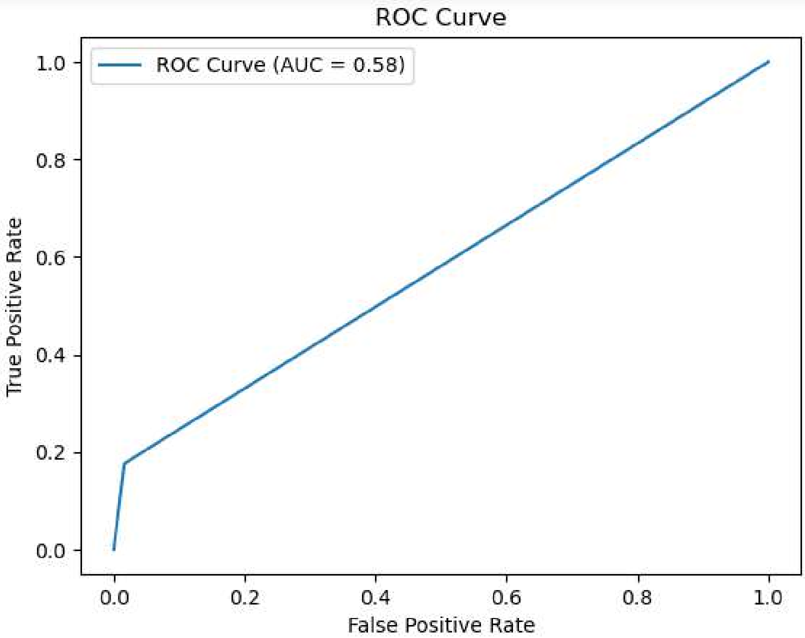




In [45]:

```
print("Decision Tree")
evaluate_model(dec_tree, X_test, y_test)
```

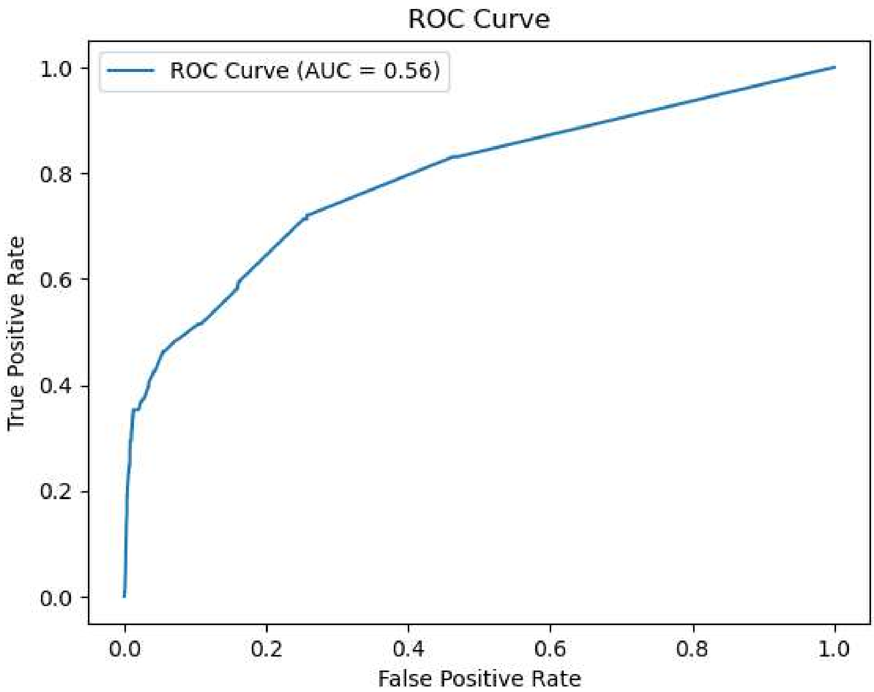
```
Decision Tree
Accuracy: 0.97
Confusion Matrix:
[[8480  134]
 [ 113   23]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      8614
           1       0.15      0.17      0.16       136

    accuracy                           0.97      8750
   macro avg       0.57      0.58      0.57      8750
```



In [95]:
```
print("Random Forest")
evaluate_model(rand_forest, X_test, y_test)
```

```
Random Forest
Accuracy: 0.98
Confusion Matrix:
[[8587   27]
 [ 118   18]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      8614
           1       0.40      0.13      0.20       136

    accuracy                           0.98      8750
   macro avg       0.69      0.56      0.60      8750
```

```
Accuracy: 0.98
Confusion Matrix:
[[8587   27]
 [ 118   18]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      8614
           1       0.40      0.13      0.20       136

    accuracy                           0.98      8750
   macro avg       0.69      0.56      0.60      8750
weighted avg       0.98      0.98      0.98      8750

ROC AUC Score: 0.56
```



In [10]:
```
# Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
```

In [ ]:

In [4]:
```
df = pd.read_csv('train_data.csv')
df
```

Out[4]:

	ID	Gender	Has a car	Has a property	Children count	Income	Employment status	Education level	Marital status	Dwelling	Age	Employment length	Has a mobile phone	Has a work phone	Has a phone	Has an email
0	5037048	M	Y	Y	0	135000.0	Working	Secondary / secondary special	Married	With parents	-16271	-3111	1	0	0	0
1	5044630	F	Y	N	1	135000.0	Commercial associate	Higher education	Single / not	House / apartment	-10130	-1651	1	0	0	0