ml-project

December 4, 2023

[20]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
```

[22]:
```
pip install --upgrade scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (1.2.2)
Collecting scikit-learn
  Downloading
scikit_learn-1.3.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(10.8 MB)
                           10.8/10.8 MB
23.7 MB/s eta 0:00:00
Requirement already satisfied: numpy<2.0,>=1.17.3 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
Installing collected packages: scikit-learn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.2.2
    Uninstalling scikit-learn-1.2.2:
      Successfully uninstalled scikit-learn-1.2.2
Successfully installed scikit-learn-1.3.2
```

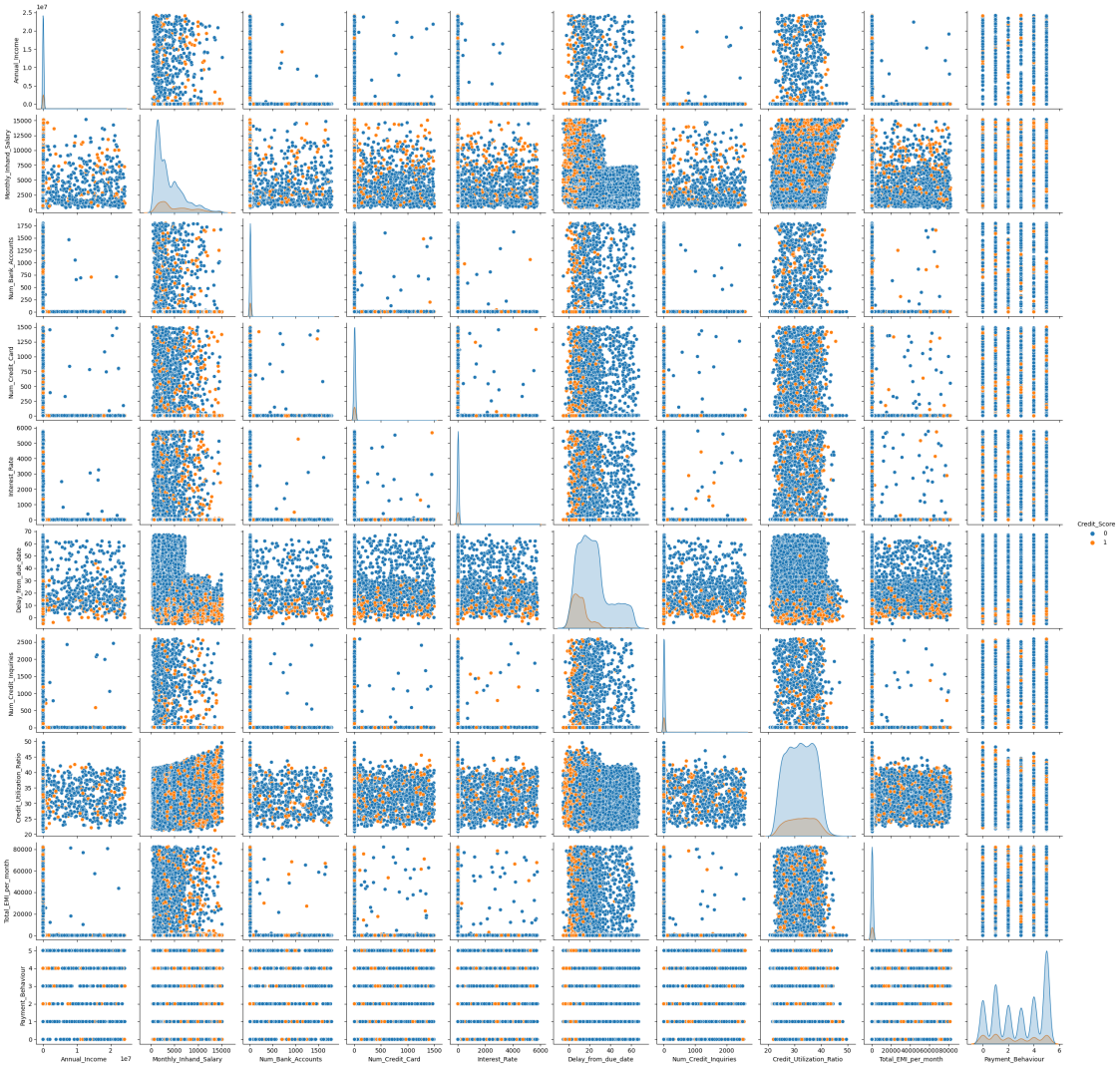
[18]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, roc_auc_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
#import scikitplot as skplt
import numpy as np
```

[7]:
```
# Step 1: Load and preprocess the data
data = pd.read_csv('/content/train.csv')
data.dropna(inplace=True)
label_encoder = LabelEncoder()
data['Payment_Behaviour'] = label_encoder.
 ↪fit_transform(data['Payment_Behaviour'])
features = data[['Annual_Income', 'Monthly_Inhand_Salary', 'Num_Bank_Accounts',␣
 ↪'Num_Credit_Card', 'Payment_Behaviour']]
target = data['Credit_Score']
```

```
<ipython-input-7-f4bd4b437412>:2: DtypeWarning: Columns (26) have mixed types.
Specify dtype option on import or set low_memory=False.
  data = pd.read_csv('/content/train.csv')
```

[8]:
```
# Step 2: Perform Exploratory Data Analysis (EDA)
plt.figure(figsize=(12, 8))
sns.pairplot(data, hue='Credit_Score', diag_kind='kde')
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



[9]:
```
X_train, X_test, y_train, y_test = train_test_split(features, target,␣
↪test_size=0.2, random_state=42)
```

[10]:
```
# Step 4: Train the models
# Logistic Regression
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
```

[10]:
```
LogisticRegression()
```

[11]:
```
# Random Forest
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
```

[11]:
```
RandomForestClassifier()
```

[12]:
```
# Gradient Boosting
gb_model = GradientBoostingClassifier()
gb_model.fit(X_train, y_train)
```

[12]:
```
GradientBoostingClassifier()
```

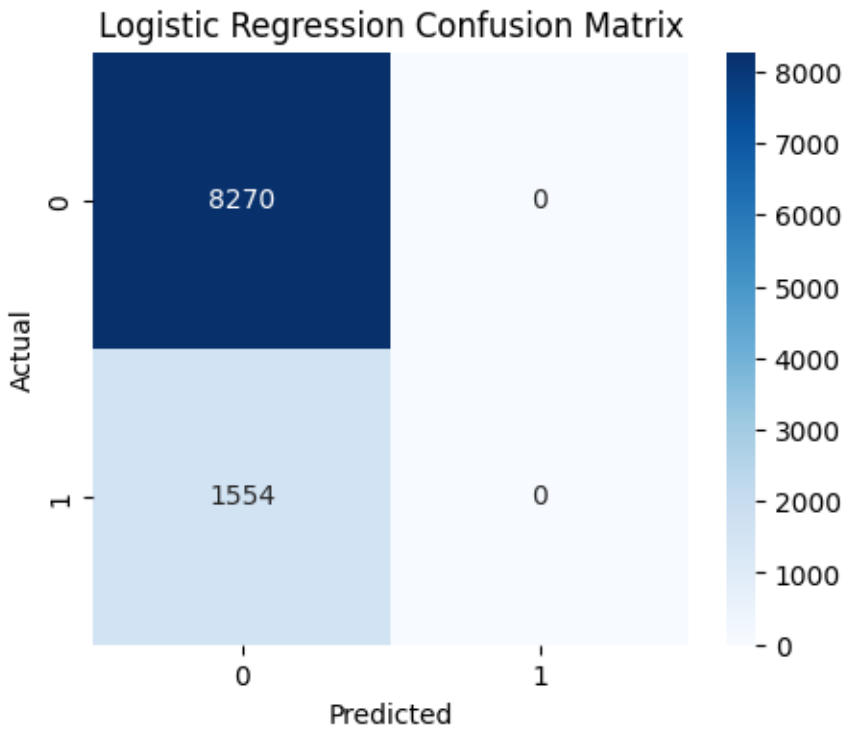
[13]:
```
# Support Vector Machine (SVM)
svm_model = SVC(probability=True)
svm_model.fit(X_train, y_train)
```

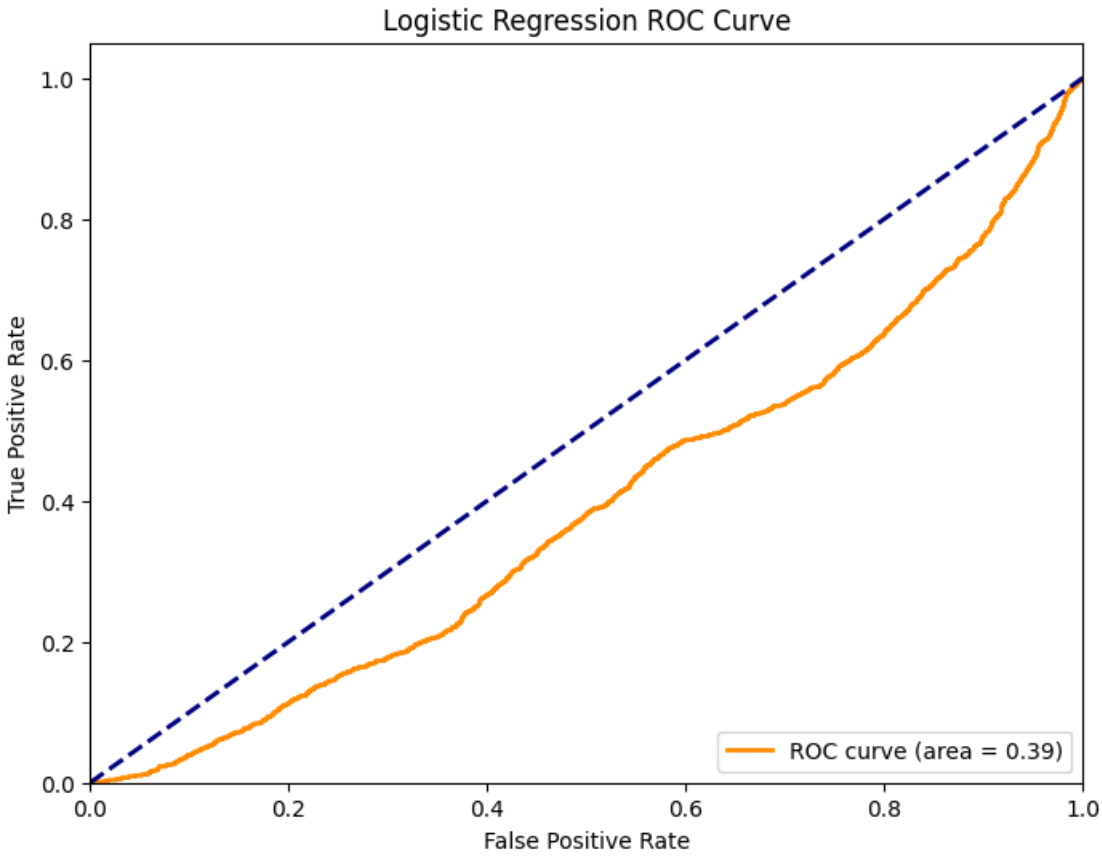
[13]:
```
SVC(probability=True)
```

[24]:
```
# Step 5: Evaluate the models
models = [logistic_model, rf_model, gb_model, svm_model]
model_names = ['Logistic Regression', 'Random Forest', 'Gradient Boosting',␣
 ↪'Support Vector Machine']

for model, name in zip(models, model_names):
    predictions = model.predict(X_test)
    probabilities = model.predict_proba(X_test)[:, 1]

    # Model evaluation metrics
    accuracy = accuracy_score(y_test, predictions)
    precision = precision_score(y_test, predictions, zero_division=1)
    recall = recall_score(y_test, predictions)
    f1 = f1_score(y_test, predictions)
    roc_auc = roc_auc_score(y_test, probabilities)

    print(f"\n{name} Metrics:")
    print(f"Accuracy: {accuracy}")
    print(f"Precision: {precision}")
    print(f"Recall: {recall}")
    print(f"F1-Score: {f1}")
    print(f"ROC-AUC: {roc_auc}")

    # Confusion Matrix
    cm = confusion_matrix(y_test, predictions)
    plt.figure(figsize=(5, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['0', '1'],␣
 ↪yticklabels=['0', '1'])
    plt.title(f'{name} Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```

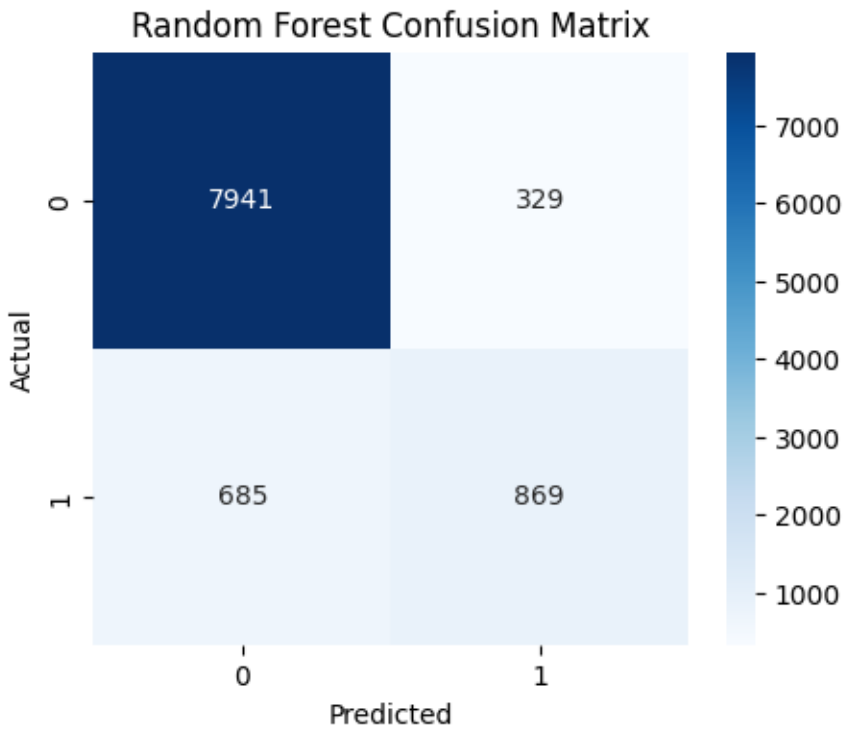
```
    # ROC Curve
    # Alternative ROC Curve (for older scikit-learn versions)
    from sklearn.metrics import roc_curve, auc
    # Get ROC curve
    fpr, tpr, thresholds = roc_curve(y_test, model.predict_proba(X_test)[:, 1])
    # Calculate AUC
    roc_auc = auc(fpr, tpr)
    # Plot ROC curve
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =␣
↪{roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'{name} ROC Curve')
    plt.legend(loc="lower right")
    plt.show()
```

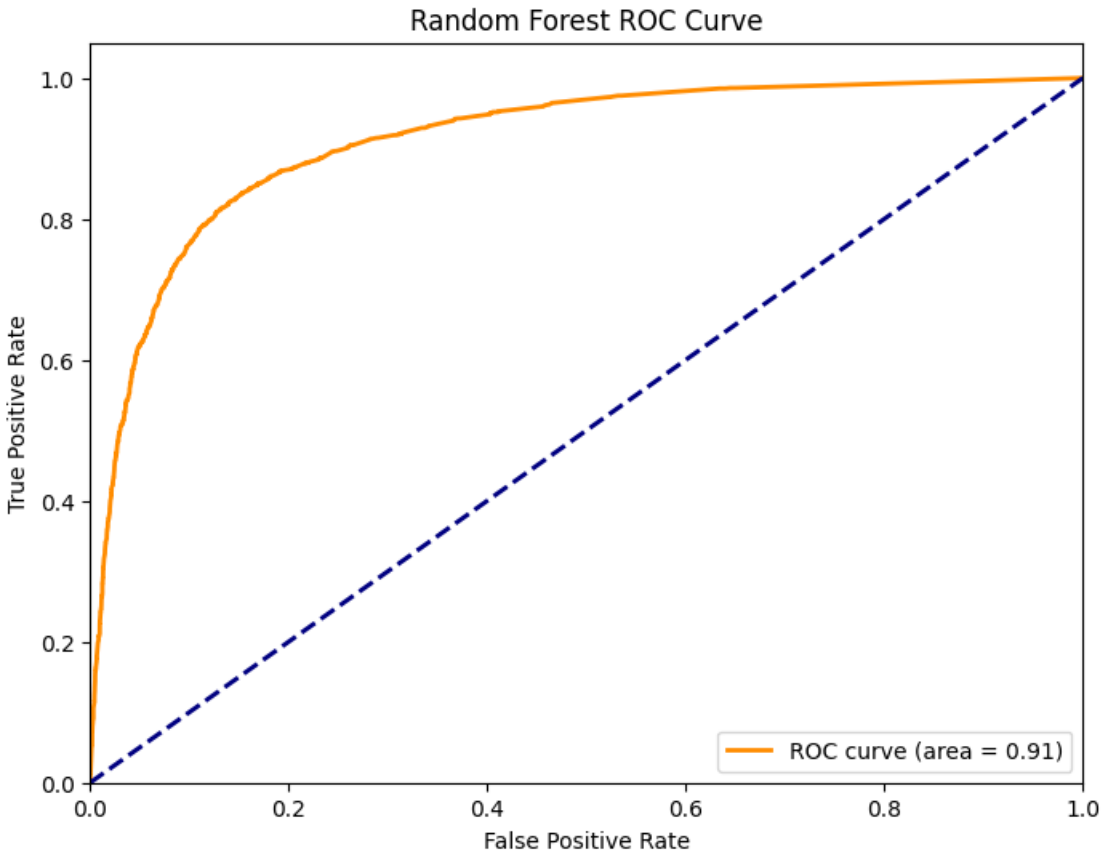
```
Logistic Regression Metrics:
Accuracy: 0.8418159609120521
Precision: 1.0
Recall: 0.0
F1-Score: 0.0
ROC-AUC: 0.3893113142508548
```


Logistic Regression Confusion Matrix
8270
0
1554
0
Actual
0
1
Predicted
8000
7000
6000
5000
4000
3000
2000
1000
0

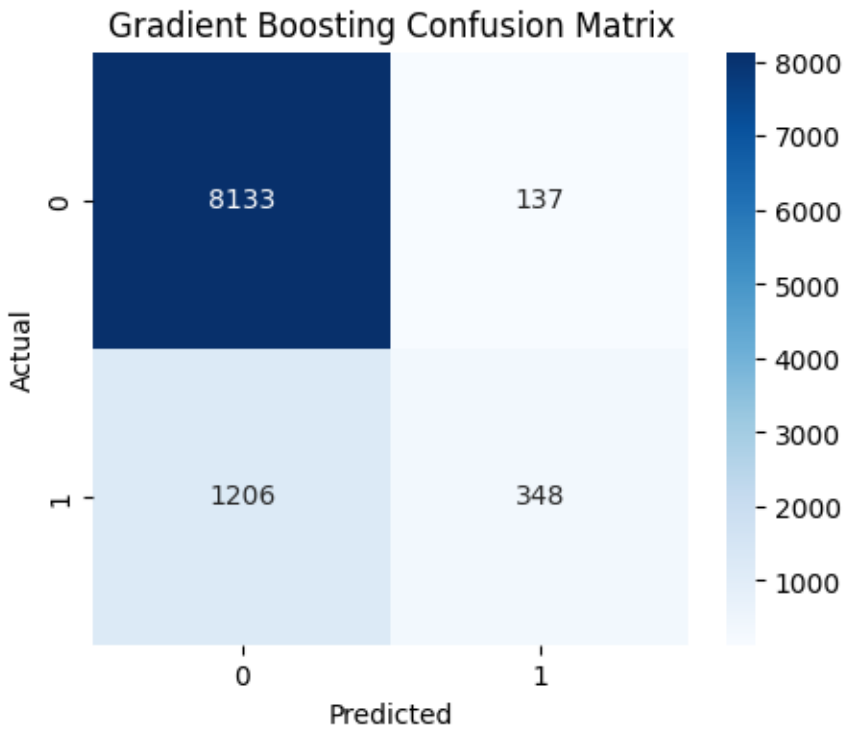


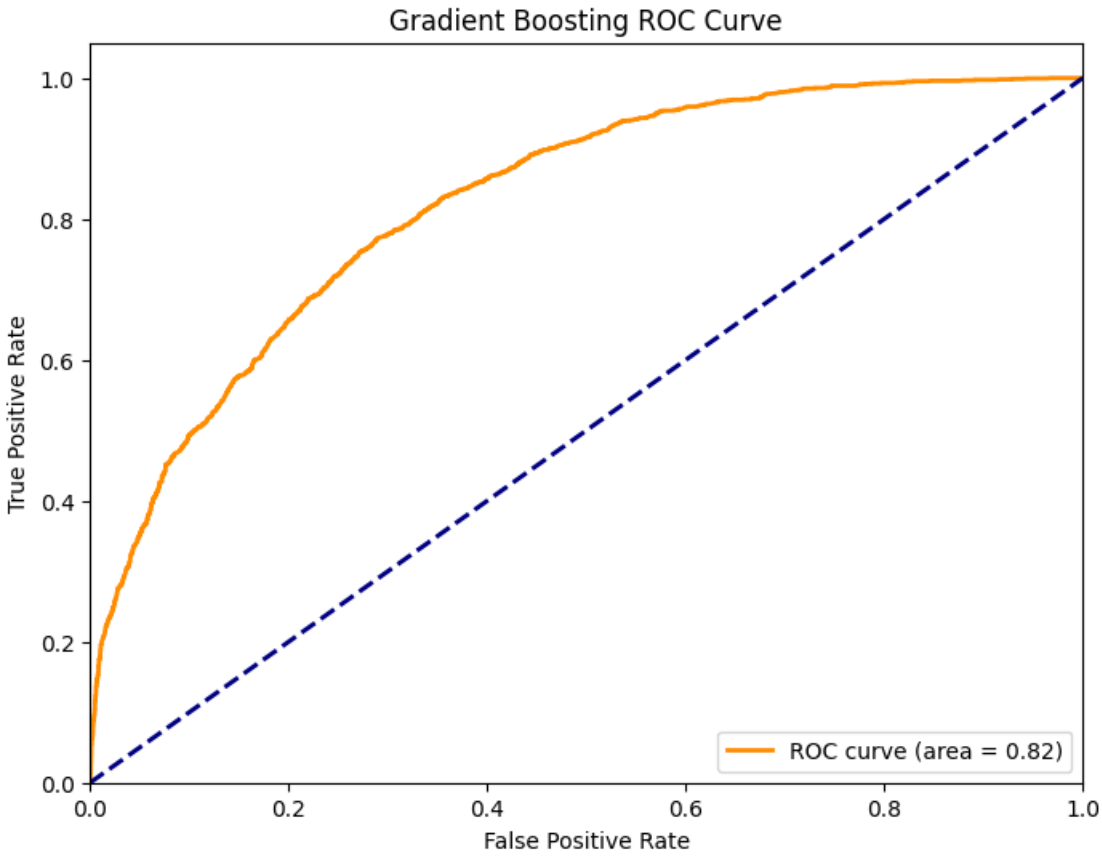

```
Random Forest Metrics:
Accuracy: 0.8967833876221498
Precision: 0.7253756260434057
Recall: 0.5592020592020592
F1-Score: 0.6315406976744187
ROC-AUC: 0.9114326020613809
```


Random Forest Confusion Matrix
7941
329
685
869
0
1
Actual
0
1
Predicted
7000
6000
5000
4000
3000
2000
1000

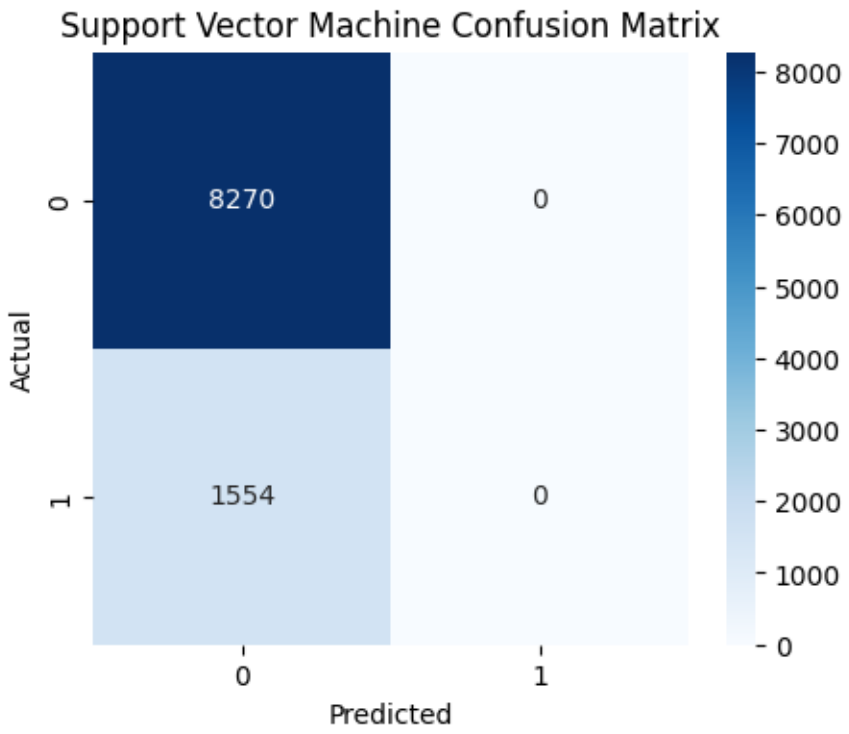


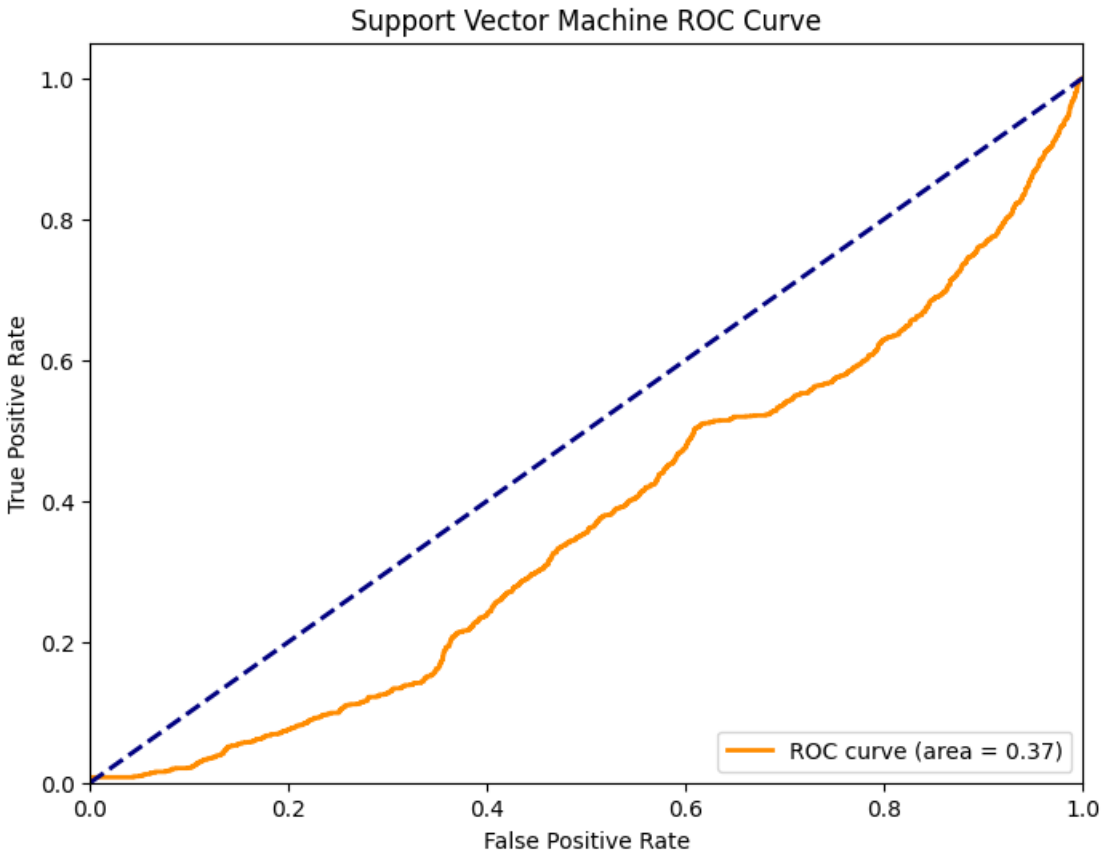

```
Gradient Boosting Metrics:
Accuracy: 0.8632939739413681
Precision: 0.7175257731958763
Recall: 0.22393822393822393
F1-Score: 0.34134379597842074
ROC-AUC: 0.8231382055747232
```


Gradient Boosting Confusion Matrix
8133
137
1206
348
Actual
0
1
Predicted
0
1
8000
7000
6000
5000
4000
3000
2000
1000




```
Support Vector Machine Metrics:
Accuracy: 0.8418159609120521
Precision: 1.0
Recall: 0.0
F1-Score: 0.0
ROC-AUC: 0.37001952289134876
```


Support Vector Machine Confusion Matrix
8270
0
1554
0
Actual
0
1
Predicted
8000
7000
6000
5000
4000
3000
2000
1000
0



[25]:
```
# Step 6: Add a column for predicted Credit Worthiness to the original dataset␣
↪using Random Forest model
data['Credit_Worthiness'] = rf_model.predict(features)
```

[26]:
```
data.to_csv('train_with_credit_worthiness.csv', index=False)
```