Identifying Stable vs Unstable Relationships Using Classical Machine Learning Techniques

Importing Libraries

In [12]:
```python
# SECTION 1: Import Required Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# To split dataset and build ML model
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings('ignore')
```

In [13]:
```python
# Loading Dataset
data=pd.read_csv("relationship_stability_dataset.csv")
data
```

Out[13]:

	communication_frequency	average_reply_time_minutes	initiates_conversation_percent	conflict_frequency	apology_sincerity_score	support_in_stress_score	respect_boundaries_score	makes_future_plans_score	jealousy_level_score	relationship_stability
0	7	70	41	0	7	8	8	9	1	1
1	15	252	59	8	10	7	3	5	1	0
2	11	275	16	7	9	5	4	10	5	0
3	8	182	17	1	7	10	10	4	9	0
4	7	167	74	7	1	10	6	5	6	0
...	...	...	...	...	...	...	...	...	...	...
495	17	288	24	5	7	4	4	10	10	0
496	7	266	43	9	6	1	8	1	2	0
497	13	162	70	3	2	2	2	10	1	0
498	4	87	44	4	8	5	6	10	4	0
499	4	55	73	8	4	8	6	7	4	0

500 rows × 10 columns

In [14]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 10 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   communication_frequency         500 non-null    int64
 1   average_reply_time_minutes      500 non-null    int64
 2   initiates_conversation_percent  500 non-null    int64
 3   conflict_frequency              500 non-null    int64
 4   apology_sincerity_score         500 non-null    int64
 5   support_in_stress_score         500 non-null    int64
 6   respect_boundaries_score        500 non-null    int64
 7   makes_future_plans_score        500 non-null    int64
 8   jealousy_level_score            500 non-null    int64
 9   relationship_stability          500 non-null    int64
dtypes: int64(10)
memory usage: 39.2 KB
```

In [15]:
```python
data.describe()
```

Out[15]:

	communication_frequency	average_reply_time_minutes	initiates_conversation_percent	conflict_frequency	apology_sincerity_score	support_in_stress_score	respect_boundaries_score	makes_future_plans_score	jealousy_level_score	relationship_stability
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000
mean	9.616000	146.712000	54.086000	4.514000	5.454000	5.420000	5.40400	5.466000	5.588000	0.04200
std	5.662922	86.944832	25.835509	2.898034	2.783411	2.847423	2.84411	2.906844	2.838021	0.20079
min	1.000000	1.000000	10.000000	0.000000	1.000000	1.000000	1.00000	1.000000	1.000000	0.00000
25%	4.750000	71.500000	32.000000	2.000000	3.000000	3.000000	3.00000	3.000000	3.000000	0.00000
50%	9.000000	145.000000	53.500000	5.000000	5.000000	5.000000	5.00000	5.000000	6.000000	0.00000
75%	15.000000	222.250000	76.000000	7.000000	8.000000	8.000000	8.00000	8.000000	8.000000	0.00000
max	19.000000	299.000000	99.000000	9.000000	10.000000	10.000000	10.00000	10.000000	10.000000	1.00000

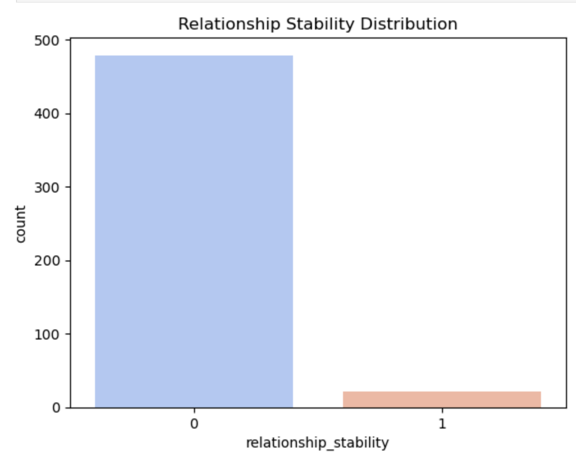
In [16]:
```python
data.head(10)
```

Out[16]:

	communication_frequency	average_reply_time_minutes	initiates_conversation_percent	conflict_frequency	apology_sincerity_score	support_in_stress_score	respect_boundaries_score	makes_future_plans_score	jealousy_level_score	relationship_stability
0	7	70	41	0	7	8	8	9	1	1
1	15	252	59	8	10	7	3	5	1	0
2	11	275	16	7	9	5	4	10	5	0
3	8	182	17	1	7	10	10	4	9	0
4	7	167	74	7	1	10	6	5	6	0
5	19	91	66	8	8	8	10	10	3	0
6	11	202	76	0	4	9	3	5	3	0
7	11	19	97	4	3	5	2	8	7	0
8	4	39	96	7	7	10	6	10	6	0
9	8	126	68	1	3	5	5	8	9	0

Check Distribution of Target Variable

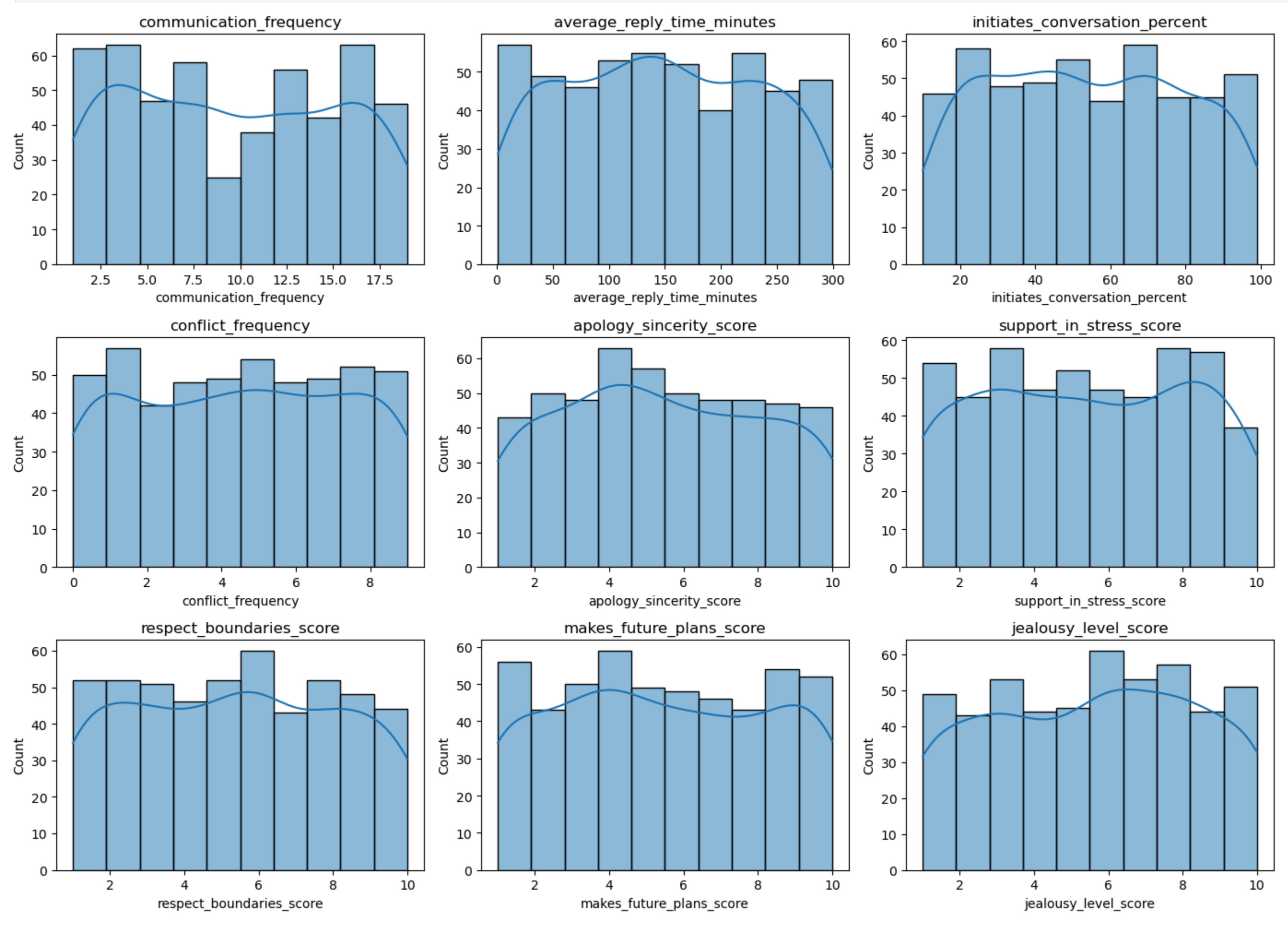
In [19]:
```python
# SECTION 4: Target Variable Distribution

sns.countplot(x="relationship_stability", data=data,palette='coolwarm')
plt.title("Relationship Stability Distribution")
plt.show()

data["relationship_stability"].value_counts(normalize=True)
```



Out[19]:
```
relationship_stability
0    0.958
1    0.042
Name: proportion, dtype: float64
```

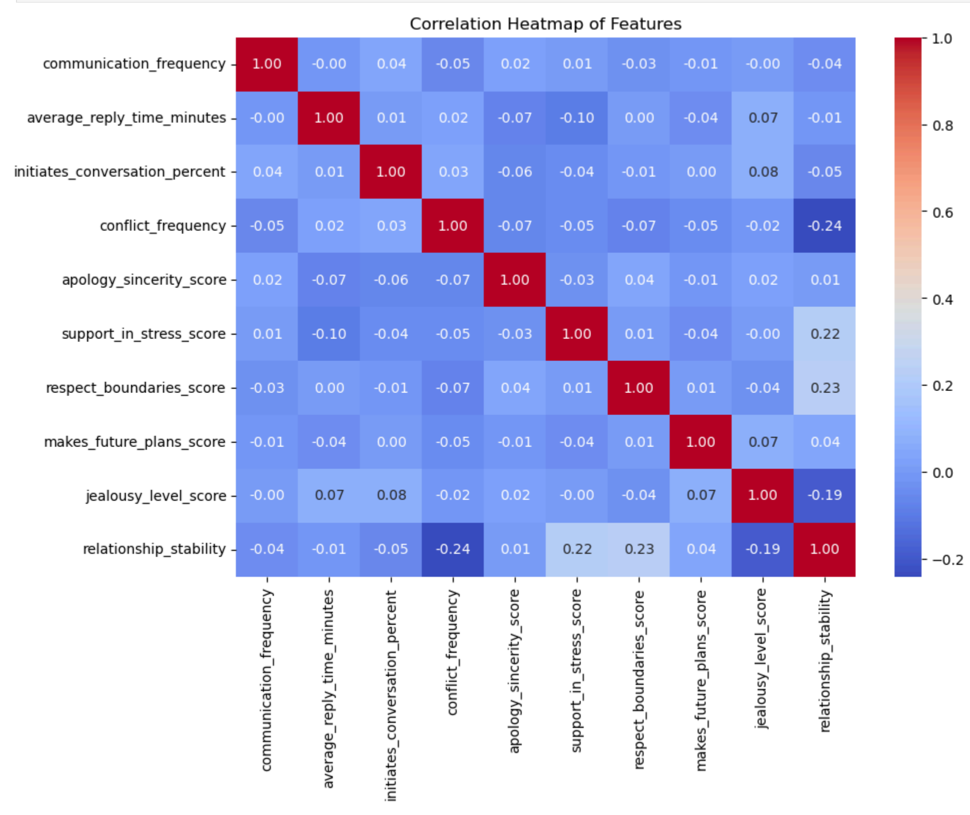
Check Feature Distributions

In [23]:
```python
#  Distribution Plots for Key Numerical Features

numeric_cols = [
    "communication_frequency",
    "average_reply_time_minutes",
    "initiates_conversation_percent",
    "conflict_frequency",
    "apology_sincerity_score",
    "support_in_stress_score",
    "respect_boundaries_score",
    "makes_future_plans_score",
    "jealousy_level_score"
]

plt.figure(figsize=(14, 10))

for i, col in enumerate(numeric_cols, 1):
    plt.subplot(3, 3, i)
    sns.histplot(data[col], kde=True,palette='Greens')
    plt.title(col)

plt.tight_layout()
plt.show()
```



Check Correlation Between Features

In [25]:
```python
# SECTION 6: Correlation Heatmap

plt.figure(figsize=(10, 7))
sns.heatmap(data.corr(), annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Heatmap of Features")
plt.show()
```



Compare Feature Means Based on Stability

In [27]:
```python
# SECTION 7: Group Comparison Based on the Target

data.groupby("relationship_stability")[numeric_cols].mean()
```

Out[27]:

relationship_stability	communication_frequency	average_reply_time_minutes	initiates_conversation_percent	conflict_frequency	apology_sincerity_score	support_in_stress_score	respect_boundaries_score	makes_future_plans_score	jealousy_level_score
0	9.663883	146.810021	54.382046	4.659708	5.448852	5.288100	5.265136	5.440501	5.701461
1	8.523810	144.476190	47.333333	1.190476	5.571429	8.428571	8.571429	6.047619	3.000000

Train-Test Split

In [28]:
```python
# SECTION 8: Train-Test Split

X = data.drop("relationship_stability", axis=1)
y = data["relationship_stability"]

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

X_train.shape, X_test.shape
```

Out[28]:
```
((400, 9), (100, 9))
```

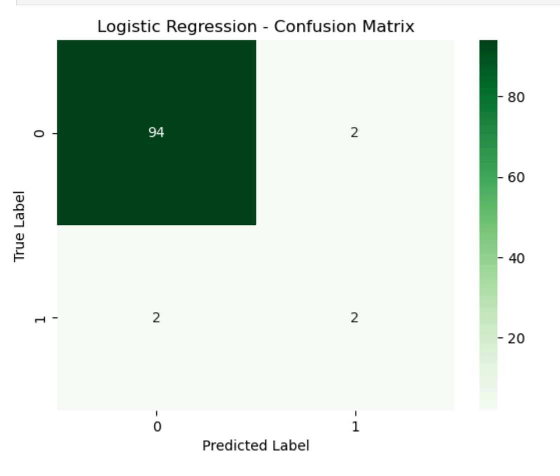
Baseline Model – Logistic Regression

In [29]:
```python
#  Logistic Regression Model

log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)

y_pred_log = log_model.predict(X_test)

print("Accuracy (Logistic Regression):", accuracy_score(y_test, y_pred_log))
print("\nClassification Report:\n", classification_report(y_test, y_pred_log))
```

```
Accuracy (Logistic Regression): 0.96

Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.98      0.98        96
           1       0.50      0.50      0.50         4

    accuracy                           0.96       100
   macro avg       0.74      0.74      0.74       100
weighted avg       0.96      0.96      0.96       100
```

Confusion Matrix for Logistic Regression

In [32]:
```python
# SECTION 10: Confusion Matrix for Logistic Regression

cm = confusion_matrix(y_test, y_pred_log)

sns.heatmap(cm, annot=True, fmt='d', cmap='Greens')
plt.title("Logistic Regression - Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



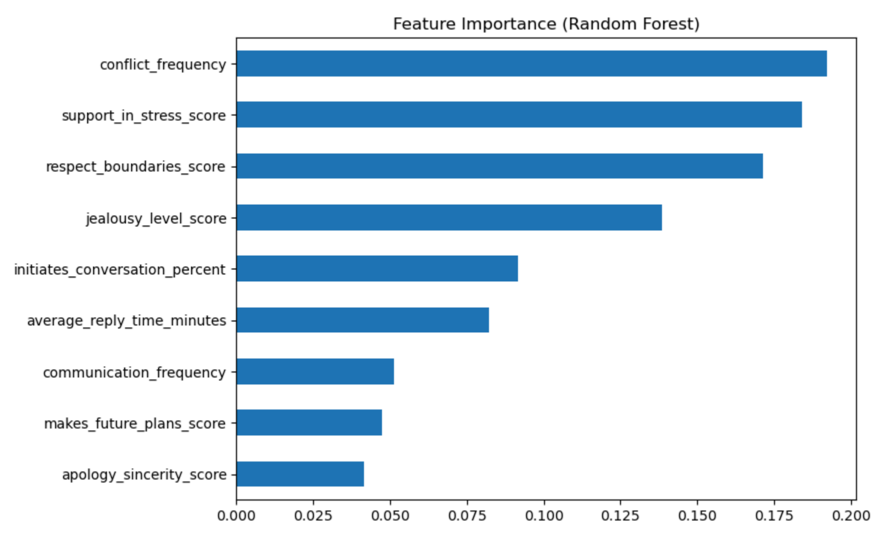
Random Forest Model (Main Model)

In [33]:
```python
# SECTION 11: Random Forest Classifier

rf_model = RandomForestClassifier(
    n_estimators=300,
    max_depth=None,
    random_state=42
)

rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

print("Accuracy (Random Forest):", accuracy_score(y_test, y_pred_rf))
print("\nClassification Report:\n", classification_report(y_test, y_pred_rf))
```

```
Accuracy (Random Forest): 0.96

Classification Report:
               precision    recall  f1-score   support

           0       0.96      1.00      0.98        96
           1       0.00      0.00      0.00         4

    accuracy                           0.96       100
   macro avg       0.48      0.50      0.49       100
weighted avg       0.92      0.96      0.94       100
```

Feature Importance (Interpretation)

In [34]:
```python
#  Feature Importance

importances = pd.Series(rf_model.feature_importances_, index=X.columns)
importances.sort_values(ascending=True).plot(kind='barh', figsize=(8,6))
plt.title("Feature Importance (Random Forest)")
plt.show()
```


Feature Importance (Random Forest)
conflict_frequency
support_in_stress_score
respect_boundaries_score
jealousy_level_score
initiates_conversation_percent
average_reply_time_minutes
communication_frequency
makes_future_plans_score
apology_sincerity_score
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200

In [ ]: