In [ ]:
```
# Problem :
# Part 1: Intelligent Lead Scoring & CLM Prediction (Python/Scala)
# Objective: Predict lead conversion and manage the full client lifecycle using AI.

# Tasks:
# - Build a supervised ML model to score leads based on various engagement metrics.
# - Build a second model to predict customer churn or next best action (NBA).
# -Train, evaluate, and deploy via REST API (Flask/FastAPI).

# Bonus:
# - Output lifecycle stage prediction.
# - Visualize data.
# - Use Spark/Scala if needed
```

In [1]:
```
#Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import seaborn as sns

# Create Dataframe
My_Data= {
    'lead_source': ['Website', 'Email', 'Phone', 'Website', 'Email', 'Phone'] * 20,
    'engagement_score': [75, 45, 60, 80, 30, 55] * 20,
    'pages_visited': [5, 2, 3, 6, 1, 4] * 20,
    'time_spent': [300, 120, 200, 400, 90, 250] * 20,
    'converted': [1, 0, 1, 1, 0, 0] * 20,
    'churned': [0, 1, 0, 0, 1, 1] * 20
}

Data = pd.DataFrame(My_Data)

# Convert the datatype of Lead_source
Data['lead_source'] = Data['lead_source'].astype('category').cat.codes

# Create Target column for Lead_Score using existing column in dataset
X_lead = Data[['lead_source', 'engagement_score', 'pages_visited', 'time_spent']]
y_lead = Data['converted']

# create Target column for churn prediction using existing columns in dataset
X_churn = Data[['lead_source', 'engagement_score', 'pages_visited', 'time_spent']]
y_churn = Data['churned']

# Split the datasets
X_train_lead, X_test_lead, y_train_lead, y_test_lead = train_test_split(X_lead, y_l
X_train_churn, X_test_churn, y_train_churn, y_test_churn = train_test_split(X_churn

# Train lead scoring model using Random Forest
lead_model = RandomForestClassifier(random_state=42)
lead_model.fit(X_train_lead, y_train_lead)

# Train churn prediction model
```

```
churn_model = RandomForestClassifier(random_state=42)
churn_model.fit(X_train_churn, y_train_churn)

# Evaluate models
lead_preds = lead_model.predict(X_test_lead)
churn_preds = churn_model.predict(X_test_churn)

lead_report = classification_report(y_test_lead, lead_preds, output_dict=True)
churn_report = classification_report(y_test_churn, churn_preds, output_dict=True)

lead_report, churn_report
```

Out[1]:

```
({'0': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 16.0},
  '1': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 8.0},
  'accuracy': 1.0,
  'macro avg': {'precision': 1.0,
   'recall': 1.0,
   'f1-score': 1.0,
   'support': 24.0},
  'weighted avg': {'precision': 1.0,
   'recall': 1.0,
   'f1-score': 1.0,
   'support': 24.0}},
 {'0': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 8.0},
  '1': {'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 16.0},
  'accuracy': 1.0,
  'macro avg': {'precision': 1.0,
   'recall': 1.0,
   'f1-score': 1.0,
   'support': 24.0},
  'weighted avg': {'precision': 1.0,
   'recall': 1.0,
   'f1-score': 1.0,
   'support': 24.0}})
```

In [3]:

```
Data
```

Out[3]:

	lead_source	engagement_score	pages_visited	time_spent	converted	churned
0	2	75	5	300	1	0
1	0	45	2	120	0	1
2	1	60	3	200	1	0
3	2	80	6	400	1	0
4	0	30	1	90	0	1
...	...	...	...	...	...	...
115	0	45	2	120	0	1
116	1	60	3	200	1	0
117	2	80	6	400	1	0
118	0	30	1	90	0	1
119	1	55	4	250	0	1

120 rows × 6 columns

In [4]:
```
pip install flask scikit-learn pandas matplotlib seaborn
```

```
Requirement already satisfied: flask in c:\users\dell\anaconda3\lib\site-packages
(3.0.3)
Requirement already satisfied: scikit-learn in c:\users\dell\anaconda3\lib\site-pack
ages (1.5.1)
Requirement already satisfied: pandas in c:\users\dell\anaconda3\lib\site-packages
(2.2.2)
Requirement already satisfied: matplotlib in c:\users\dell\anaconda3\lib\site-packag
es (3.9.2)
Requirement already satisfied: seaborn in c:\users\dell\anaconda3\lib\site-packages
(0.13.2)
Requirement already satisfied: Werkzeug>=3.0.0 in c:\users\dell\anaconda3\lib\site-p
ackages (from flask) (3.0.3)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\dell\anaconda3\lib\site-pac
kages (from flask) (3.1.4)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\dell\anaconda3\lib\si
te-packages (from flask) (2.2.0)
Requirement already satisfied: click>=8.1.3 in c:\users\dell\anaconda3\lib\site-pack
ages (from flask) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\dell\anaconda3\lib\site-pa
ckages (from flask) (1.6.2)
Requirement already satisfied: numpy>=1.19.5 in c:\users\dell\anaconda3\lib\site-pac
kages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in c:\users\dell\anaconda3\lib\site-pack
ages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\dell\anaconda3\lib\site-pac
kages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\dell\anaconda3\lib\s
ite-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\dell\anaconda3\lib
\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\anaconda3\lib\site-pack
ages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\dell\anaconda3\lib\site-pa
ckages (from pandas) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\dell\anaconda3\lib\site-
packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\dell\anaconda3\lib\site-pack
ages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\dell\anaconda3\lib\site
-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\dell\anaconda3\lib\site
-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\dell\anaconda3\lib\site-p
ackages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\dell\anaconda3\lib\site-package
s (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\dell\anaconda3\lib\site-
packages (from matplotlib) (3.1.2)
Requirement already satisfied: colorama in c:\users\dell\anaconda3\lib\site-packages
(from click>=8.1.3->flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\dell\anaconda3\lib\site-p
ackages (from Jinja2>=3.1.2->flask) (2.1.3)
Requirement already satisfied: six>=1.5 in c:\users\dell\anaconda3\lib\site-packages
(from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from flask import Flask, request, jsonify
import joblib

# Dummy data generation
def generate_data():
    data = {
        'lead_source': ['Website', 'Email', 'Phone'] * 40,
        'engagement_score': [75, 45, 60] * 40,
        'pages_visited': [5, 2, 3] * 40,
        'time_spent': [300, 120, 200] * 40,
        'converted': [1, 0, 1] * 40,
        'churned': [0, 1, 0] * 40
    }
    df = pd.DataFrame(data)
    df['lead_source'] = df['lead_source'].astype('category').cat.codes
    return df

df = generate_data()

# Train Lead Scoring Model
X_lead = df[['lead_source', 'engagement_score', 'pages_visited', 'time_spent']]
y_lead = df['converted']
X_train_lead, _, y_train_lead, _ = train_test_split(X_lead, y_lead, test_size=0.2)
lead_model = RandomForestClassifier().fit(X_train_lead, y_train_lead)
joblib.dump(lead_model, 'lead_model.pkl')

# Train Churn Prediction Model
y_churn = df['churned']
X_train_churn, _, y_train_churn, _ = train_test_split(X_lead, y_churn, test_size=0.
churn_model = RandomForestClassifier().fit(X_train_churn, y_train_churn)
joblib.dump(churn_model, 'churn_model.pkl')
```

Out[5]:
```
['churn_model.pkl']
```

In [7]:
```
from flask import Flask, request, jsonify
import joblib

app = Flask(__name__)
lead_model = joblib.load('lead_model.pkl')
churn_model = joblib.load('churn_model.pkl')

@app.route('/predict-lead', methods=['POST'])
def predict_lead():
    data = request.json
    pred = lead_model.predict([list(data.values())])[0]
    return jsonify({'lead_conversion': int(pred)})

@app.route('/predict-churn', methods=['POST'])
def predict_churn():
    data = request.json
    pred = churn_model.predict([list(data.values())])[0]
    return jsonify({'churn_prediction': int(pred)})
```

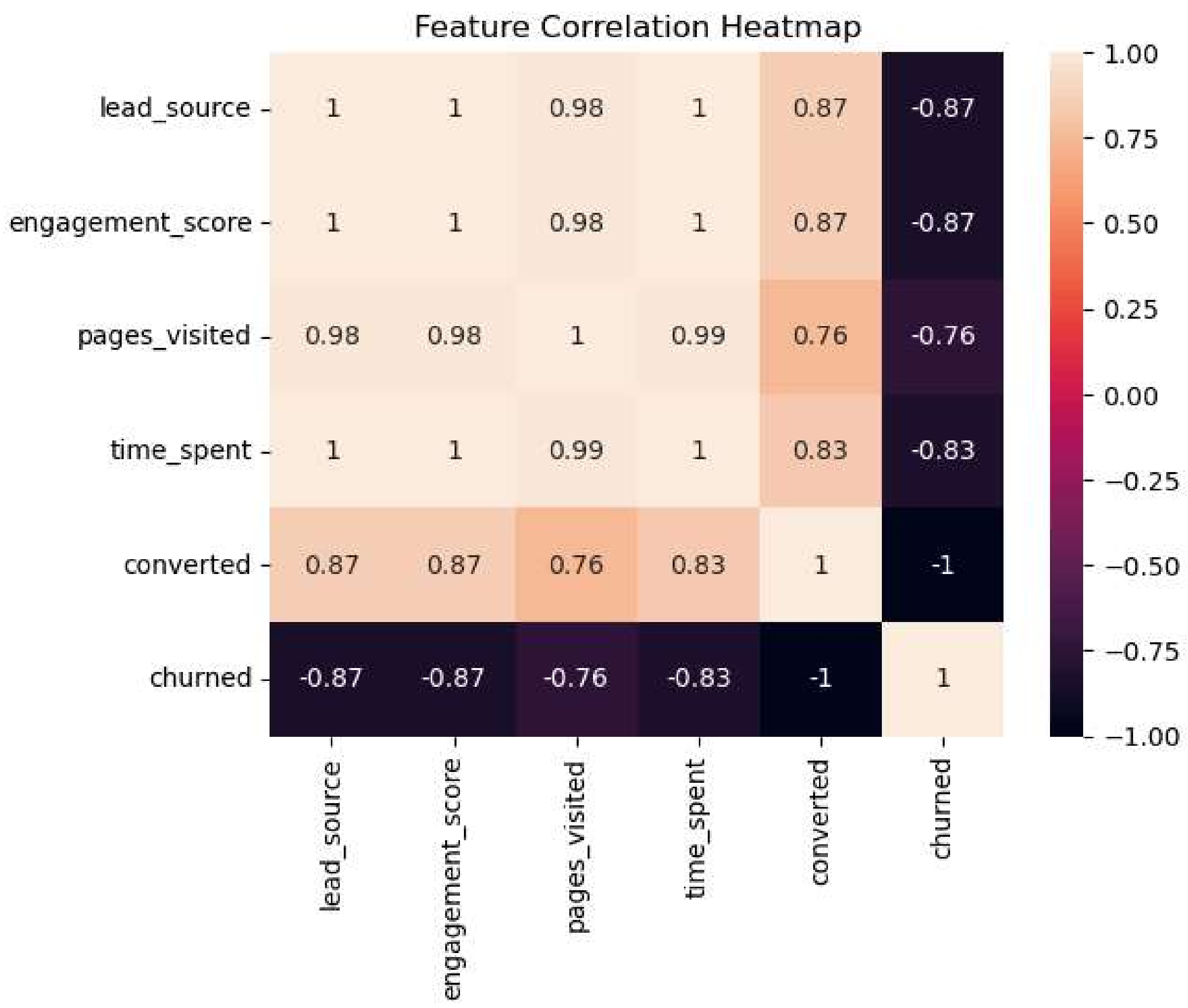
```
if __name__ == '__main__':
    app.run(debug=True)
```

```
 * Serving Flask app '__main__'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use
a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
```

In [8]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Correlation heatmap
sns.heatmap(df.corr(), annot=True)
plt.title("Feature Correlation Heatmap")
plt.show()
```


Feature Correlation Heatmap
lead_source
engagement_score
pages_visited
time_spent
converted
churned