In [1]:

```
"Alzheimer Disease predection using SVM"
```

Out[1]:

```
'Alzheimer Disease predection using SVM'
```

In [2]:

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

In [3]:

```
data = pd.read_csv(r"oasis_longitudinal.csv")
```

In [4]:

```
data.head(5)
```

Out[4]:

	Subject ID	MRI ID	Group	Visit	MR Delay	M/F	Hand	Age	EDUC	SES	MMSE	CI
0	OAS2_0001	OAS2_0001_MR1	Nondemented	1	0	M	R	87	14	2.0	27.0	(
1	OAS2_0001	OAS2_0001_MR2	Nondemented	2	457	M	R	88	14	2.0	30.0	(
2	OAS2_0002	OAS2_0002_MR1	Demented	1	0	M	R	75	12	NaN	23.0	(
3	OAS2_0002	OAS2_0002_MR2	Demented	2	560	M	R	76	12	NaN	28.0	(
4	OAS2_0002	OAS2_0002_MR3	Demented	3	1895	M	R	80	12	NaN	22.0	(

In [5]:

```
data.shape
```

Out[5]:

```
(373, 15)
```

In [6]:

```
#check null value present in dataset
data.isnull().sum()
```

Out[6]:

```
Subject ID     0
MRI ID         0
Group          0
Visit          0
MR Delay       0
M/F            0
Hand           0
Age            0
EDUC           0
SES           19
MMSE           2
CDR            0
eTIV           0
nWBV           0
ASF            0
dtype: int64
```

In [7]:

```
# Handle NaN values if needed
data = data.dropna()
data.isnull().sum()
data.shape
```

Out[7]:
```
(354, 15)
```

In [8]:
```
#Before Lable
data[['Group','Hand','M/F']].head(15)
```

Out[8]:

	Group	Hand	M/F
0	Nondemented	R	M
1	Nondemented	R	M
5	Nondemented	R	F
6	Nondemented	R	F
7	Nondemented	R	M
8	Nondemented	R	M
9	Nondemented	R	M
13	Nondemented	R	F
14	Nondemented	R	F
15	Demented	R	M
16	Demented	R	M
17	Demented	R	F
18	Demented	R	F
19	Nondemented	R	F
20	Nondemented	R	F

In [9]:
```
# Encode 'Group' using label encoding
from sklearn.preprocessing import LabelEncoder, StandardScaler
label_encoder = LabelEncoder()
data['Group'] = label_encoder.fit_transform(data['Group'])
data['Hand'] = label_encoder.fit_transform(data['Hand'])
data['M/F'] = label_encoder.fit_transform(data['M/F'])
data[['Group','Hand','M/F']].head(15)
```

Out[9]:

	Group	Hand	M/F
0	2	0	1
1	2	0	1
5	2	0	0
6	2	0	0
7	2	0	1
8	2	0	1
9	2	0	1
13	2	0	0
14	2	0	0
15	1	0	1
16	1	0	1
17	1	0	0
18	1	0	0
19	2	0	0
20	2	0	0

In [10]:

```
data.describe()
```

Out[10]:

	Group	Visit	MR Delay	M/F	Hand	Age	EDUC	SES	
count	354.000000	354.000000	354.000000	354.000000	354.0	354.000000	354.000000	354.000000	35
mean	1.432203	1.884181	601.353107	0.423729	0.0	77.033898	14.703390	2.460452	2
std	0.675078	0.925330	640.596081	0.494848	0.0	7.811808	2.895662	1.134005	
min	0.000000	1.000000	0.000000	0.000000	0.0	60.000000	6.000000	1.000000	
25%	1.000000	1.000000	0.000000	0.000000	0.0	71.000000	12.000000	2.000000	2
50%	2.000000	2.000000	559.500000	0.000000	0.0	77.000000	15.000000	2.000000	2
75%	2.000000	2.000000	882.500000	1.000000	0.0	82.000000	16.750000	3.000000	3
max	2.000000	5.000000	2639.000000	1.000000	0.0	98.000000	23.000000	5.000000	3

In [11]:

```
# Drop non-numeric columns
non_numeric_columns = ['Subject ID', 'MRI ID']
data = data.drop(non_numeric_columns, axis=1)
```

In [12]:

```
##Model selection
from sklearn.model_selection import train_test_split

X = data.drop(['Group'], axis=1)
y = data['Group']
```

In [13]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [14]:

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Create SVM model
svm_model = SVC(kernel='linear', C=1)
svm_model.fit(X_train, y_train)

# Predictions
y_pred = svm_model.predict(X_test)

# Model performance
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.8873239436619719
```

In [15]:

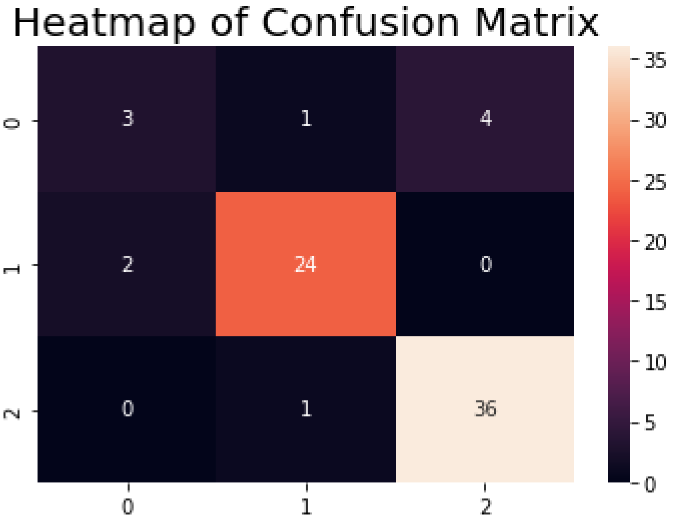
```
# Example prediction for a new data point feature by feature

# Define feature names in the same order as your dataset
# feature_names = ['Visit', 'MR Delay', 'M/F', 'Hand', 'Age', 'EDUC', 'SES', 'MMSE', '


# Initialize an empty list to store feature values
new_data_point = [1, 0, 0, 1, 75, 12, 2.0, 23.0, 0.5, 1678, 0.736, 1.046]

# # Collect values for each feature
# for feature_name in feature_names:
#     value = input(f"Enter the value for {feature_name}: ")
#     new_data_point.append(float(value))  # Ensure the input is converted to the appr

# Reshape the list to a 2D array as the model expects
new_data_point = [new_data_point]

# Map for human-readable labels
label_mapping = {2: 'Nondemented', 1: 'Demented'}

# Make a prediction
predicted_group = svm_model.predict(new_data_point)

# Map the predicted label to human-readable label
predicted_label = label_mapping[predicted_group[0]]

print(f"Predicted Group: {predicted_label}")
```

```
Predicted Group: Demented
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does n
ot have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

In [16]:

```
#confusion metrix
from sklearn.metrics import confusion_matrix,classification_report, accuracy_score
cm=confusion_matrix(y_test,y_pred)
plt.title("Heatmap of Confusion Matrix",fontsize=20)
sns.heatmap(cm,annot=True)
plt.show()
```



In [17]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.60      0.38      0.46         8
           1       0.92      0.92      0.92        26
           2       0.90      0.97      0.94        37

    accuracy                           0.89        71
   macro avg       0.81      0.76      0.77        71
weighted avg       0.87      0.89      0.88        71
```

In [ ]: