

```

In [10]: ▶ 1 import numpy as np
           2 from sklearn.neighbors import KNeighborsClassifier
           3 import matplotlib.pyplot as plt
           4
           5 # Example data: coordinates and labels (0: different route, 1: same
           6 data = np.array([
           7     [17.2404, 78.4293, 1], # Shamshabad, Hyderabad
           8     [17.5453, 78.3408, 1], # Narkuda, Hyderabad
           9     [17.1582, 78.1874, 0], # Shabad, Hyderabad
          10 ])
          11
          12 # Split data into features (X) and labels (y)
          13 X = data[:, :2]
          14 y = data[:, 2]
          15
          16 # Normalize/Scale features
          17 X_scaled = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
          18
          19 # Train KNN model
          20 k = 2 # Number of neighbors
          21 knn_model = KNeighborsClassifier(n_neighbors=k)
          22 knn_model.fit(X_scaled, y)

```

Out[10]: KNeighborsClassifier(n\_neighbors=2)

```

In [11]: ▶ 1 # Custom input Latitude and Longitude points
           2 input_points = np.array([
           3     [17.2404, 78.4293], # Shamshabad, Hyderabad
           4     [17.5453, 78.3408], # Narkuda, Hyderabad
           5     [17.7275, 78.2746], # Rajiv Gandhi International Airport (not
           6 ])

```

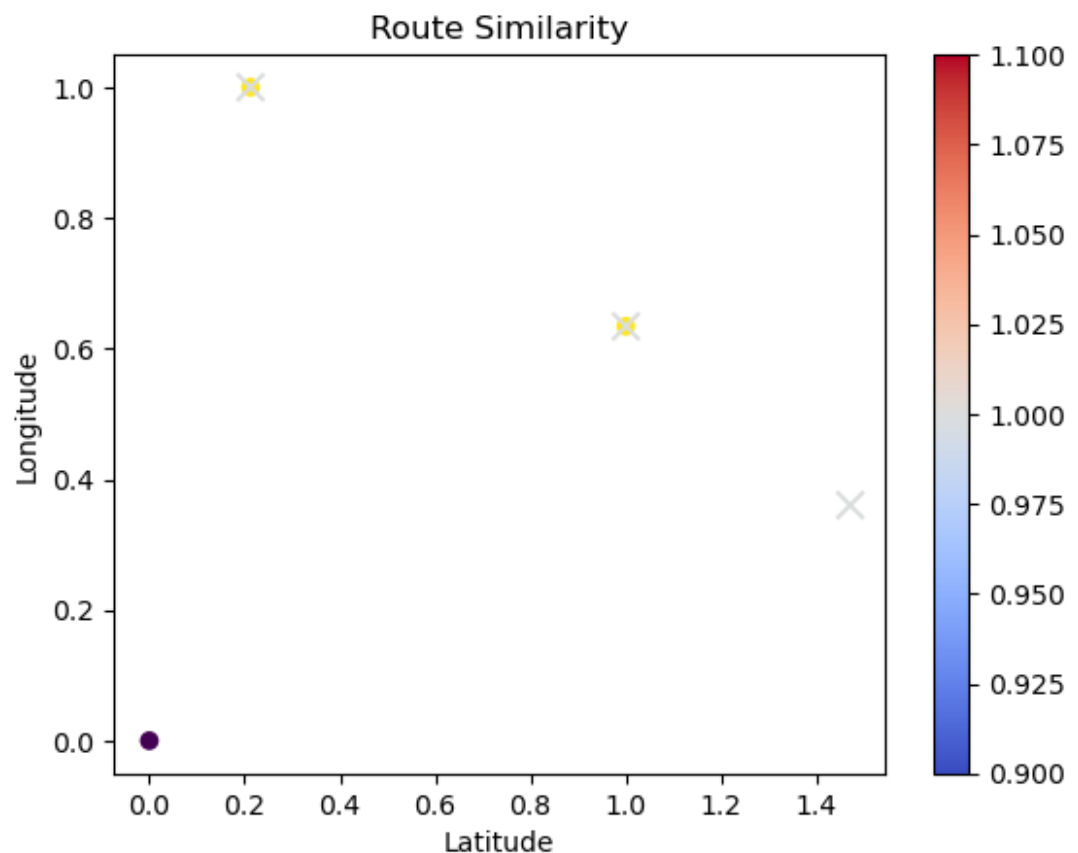
```

In [12]: ▶ 1 # Normalize/Scale input points
2 input_scaled = (input_points - X.min(axis=0)) / (X.max(axis=0) - X.
3
4 # Predict route similarity
5 predictions = knn_model.predict(input_scaled)
6
7 # Visualization
8 plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y, cmap='viridis')
9 plt.scatter(input_scaled[:, 0], input_scaled[:, 1], c=predictions,
10 plt.xlabel('Latitude')
11 plt.ylabel('Longitude')
12 plt.title('Route Similarity')
13 plt.colorbar()
14 plt.show()
15
16 print("Predictions:", predictions)

```

C:\Users\kaila\anaconda3\lib\site-packages\sklearn\neighbors\\_classification.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.

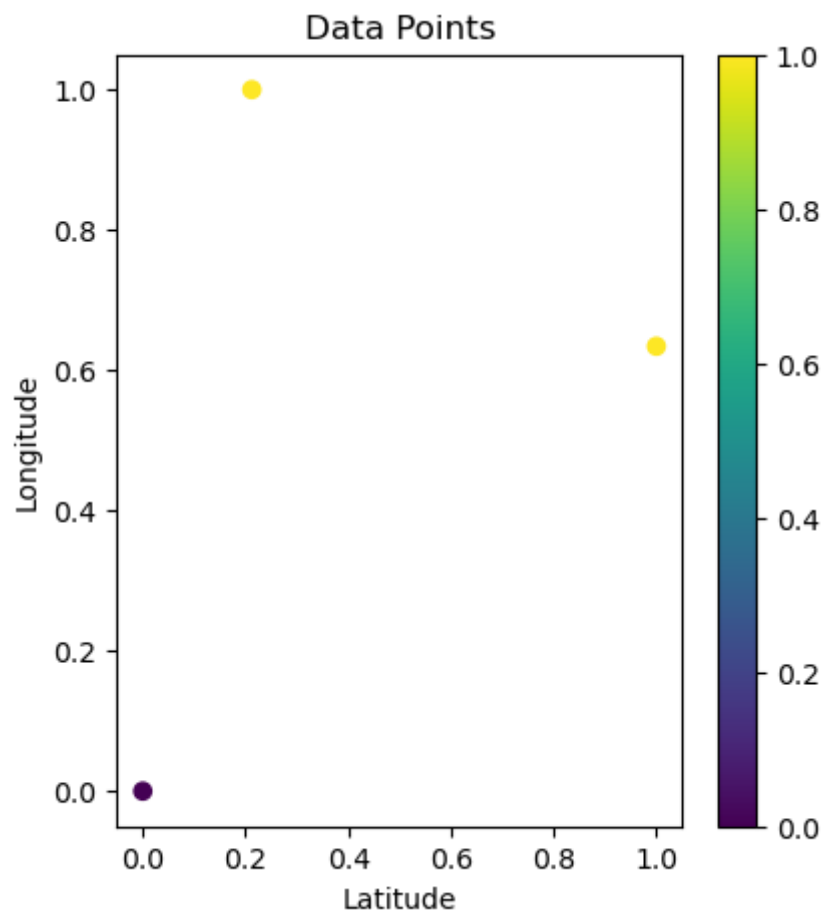
```
mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```



Predictions: [1. 1. 1.]

```
In [13]: 1 # Visualization: Scatter plot of data points
2 plt.figure(figsize=(10, 5))
3
4 plt.subplot(1, 2, 1)
5 plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y, cmap='viridis')
6 plt.xlabel('Latitude')
7 plt.ylabel('Longitude')
8 plt.title('Data Points')
9 plt.colorbar()
```

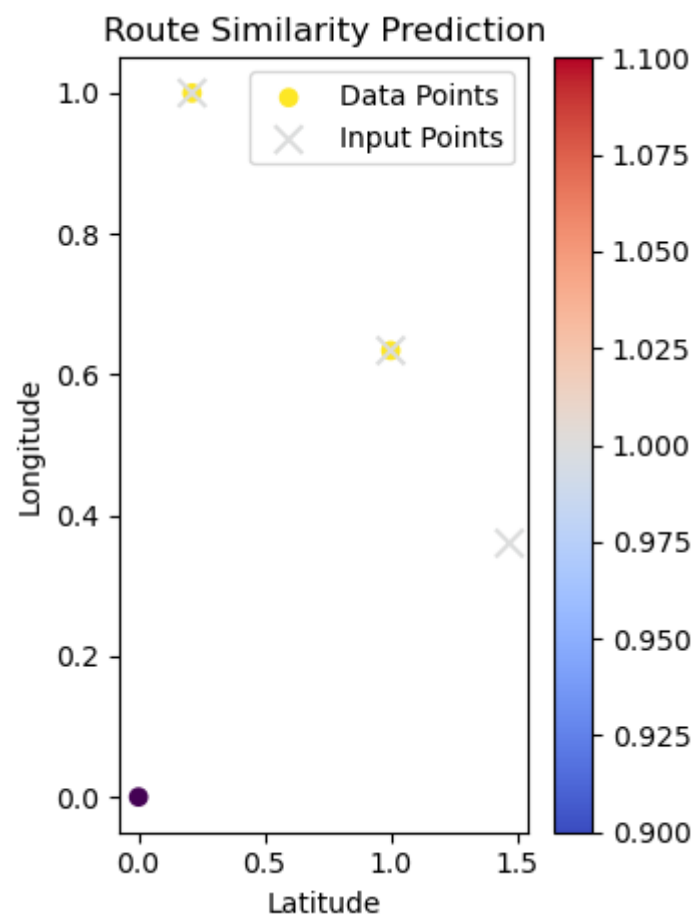
Out[13]: <matplotlib.colorbar.Colorbar at 0x1ef78e80310>



```

In [14]: 1 # Visualization: Scatter plot of input points with predictions
2 plt.subplot(1, 2, 2)
3 plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y, cmap='viridis', la
4 plt.scatter(input_scaled[:, 0], input_scaled[:, 1], c=predictions,
5 plt.xlabel('Latitude')
6 plt.ylabel('Longitude')
7 plt.title('Route Similarity Prediction')
8 plt.colorbar()
9 plt.legend()
10
11 plt.tight_layout()
12 plt.show()
13
14 print("Predictions:", predictions)
15

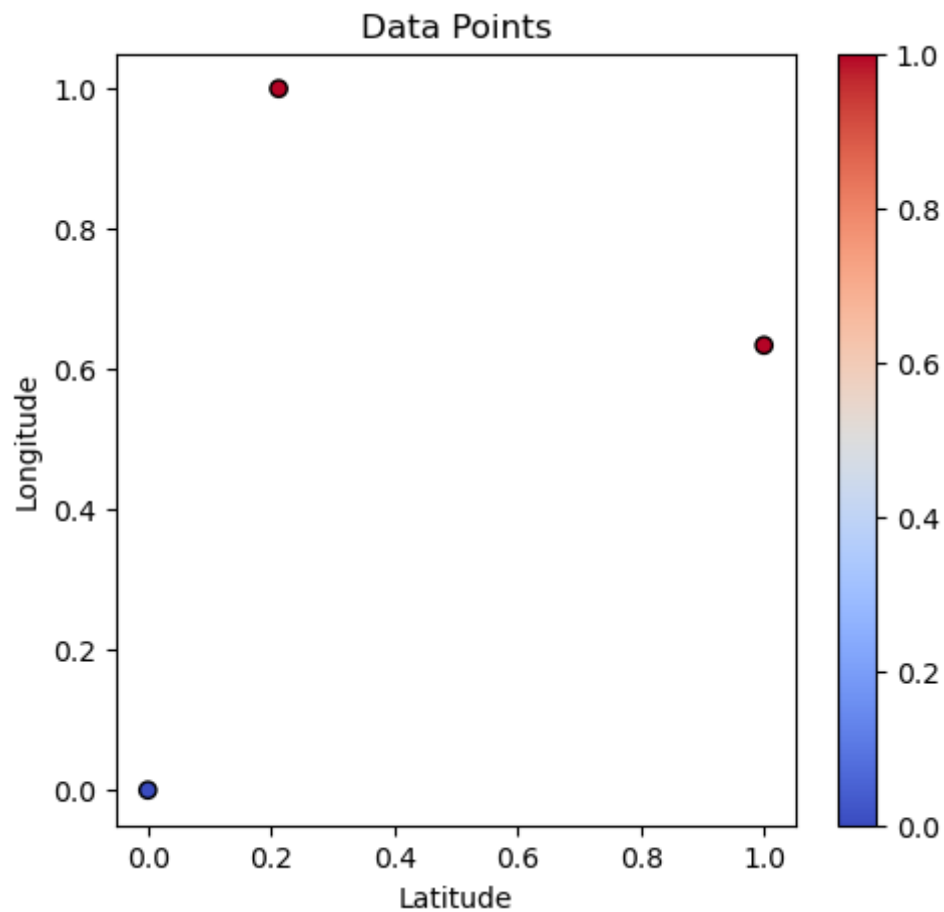
```



Predictions: [1. 1. 1.]

```
In [15]: 1 # Visualization: Scatter plot of data points
2 plt.figure(figsize=(12, 5))
3
4 plt.subplot(1, 2, 1)
5 plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y, cmap='coolwarm', e
6 plt.xlabel('Latitude')
7 plt.ylabel('Longitude')
8 plt.title('Data Points')
9 plt.colorbar()
```

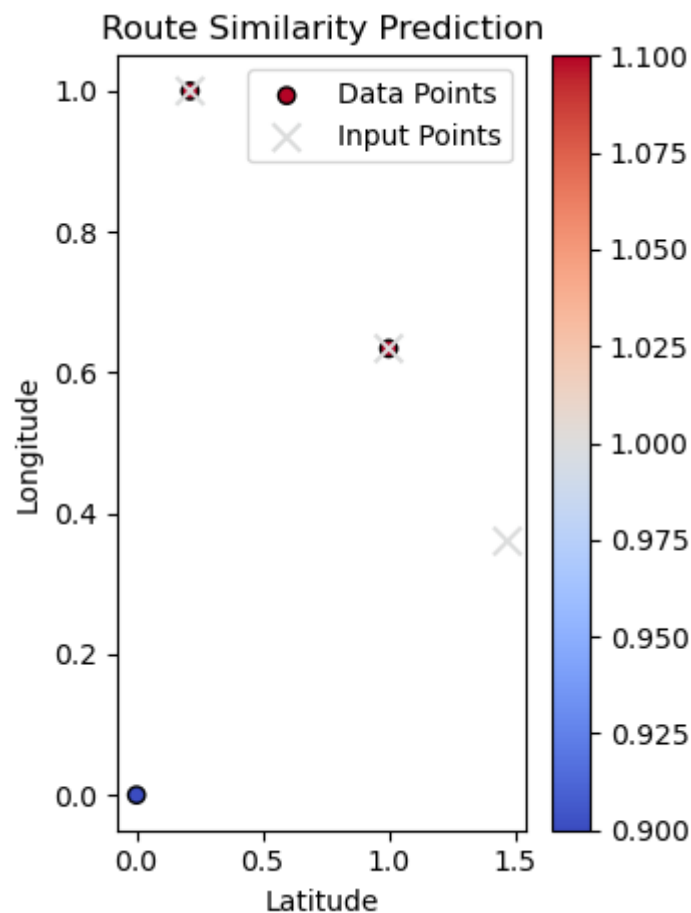
Out[15]: <matplotlib.colorbar.Colorbar at 0x1ef78fff0d0>



```

In [16]: 1 # Visualization: Scatter plot of input points with predictions
2 plt.subplot(1, 2, 2)
3 plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y, cmap='coolwarm', e
4 plt.scatter(input_scaled[:, 0], input_scaled[:, 1], c=predictions,
5 plt.xlabel('Latitude')
6 plt.ylabel('Longitude')
7 plt.title('Route Similarity Prediction')
8 plt.colorbar()
9 plt.legend()
10
11 plt.tight_layout()
12 plt.show()
13
14 print("Predictions:", predictions)
15

```



Predictions: [1. 1. 1.]

```

In [17]: 1 '''the point are taken from the API we have created by connecting t

```

Out[17]: 'the point are taken from the API we have created by connecting to LOC  
AL '



In [18]:

```
1 import numpy as np
2 from sklearn.neighbors import KNeighborsRegressor
3 from sklearn.svm import SVR
4 from sklearn.preprocessing import StandardScaler
5 from geopy.distance import geodesic
6
7 # Example data: coordinates of bus stops and travel times (in minut
8 bus_stops = np.array([
9     [17.2404, 78.4293], # Shamshabad, Hyderabad
10    [17.5453, 78.3408], # Narkuda, Hyderabad
11    [17.1582, 78.1874], # Shabad, Hyderabad
12    [17.7275, 78.2746], # Rajiv Gandhi Airport
13    [17.4265, 78.4585], # Gandipet, Hyderabad
14    [17.4646, 78.4859], # Ocean Park, Hyderabad
15    [17.4125, 78.4388], # MehdiPatnam, Hyderabad
16    [17.3986, 78.4703], # Golconda Fort, Hyderabad
17    [17.3822, 78.4867], # Qutub Shahi Tombs, Hyderabad
18    [17.3850, 78.4804], # Toli Chowki, Hyderabad
19    [17.4090, 78.4896], # Banjara Hills, Hyderabad
20    [17.4349, 78.4484], # Film Nagar, Hyderabad
21    [17.4396, 78.4170], # HITEC City, Hyderabad
22    [17.4461, 78.3490], # JNTU, Hyderabad
23    [17.4485, 78.3719], # Kukatpally, Hyderabad
24    [17.4531, 78.3636], # Moosapet, Hyderabad
25    [17.4711, 78.3656], # Bharatnagar, Hyderabad
26    [17.4861, 78.3850], # Ameerpet, Hyderabad
27    [17.5099, 78.3765], # Punjagutta, Hyderabad
28    [17.5184, 78.3893], # Nagarjuna Circle, Hyderabad
29    [17.4811, 78.4291], # Masab Tank, Hyderabad
30 ])
31
32 # Calculate distances between bus stops using geopy's geodesic
33 distances = [geodesic(bus_stops[i], bus_stops[i+1]).kilometers for
34
35 # Reshape distances for KNN input
36 X_knn = np.array(distances).reshape(-1, 1)
37 y_knn = bus_stops[1:]
38
39 # Standardize features for KNN
40 scaler_X_knn = StandardScaler()
41 X_scaled_knn = scaler_X_knn.fit_transform(X_knn)
42
43 # Train KNN model
44 knn_model = KNeighborsRegressor(n_neighbors=3)
45 knn_model.fit(X_scaled_knn, y_knn)
46
47 # Reshape distances for SVR input
48 X_svr = np.array(distances).reshape(-1, 1)
49 y_svr = travel_times[1:]
50
51 # Standardize features for SVR
52 scaler_X_svr = StandardScaler()
53 scaler_y_svr = StandardScaler()
54 X_scaled_svr = scaler_X_svr.fit_transform(X_svr)
55 y_scaled_svr = scaler_y_svr.fit_transform(y_svr.reshape(-1, 1))
56
57 # Train SVR model
58 svr_model = SVR(kernel='rbf')
59 svr_model.fit(X_scaled_svr, y_scaled_svr.ravel())
60
61 # Custom input distance
```



```

62 input_distance = geodesic(bus_stops[0], bus_stops[1]).kilometers
63
64 # Predict intermediate points using KNN
65 predicted_point = knn_model.predict([[input_distance]])
66 predicted_distance = np.array([geodesic(bus_stops[0], predicted_point)])
67
68 # Scale predicted distance for SVR
69 predicted_distance_scaled = scaler_X_svr.transform(predicted_distance_scaled.reshape(-1, 1))
70
71 # Predict travel time using SVR
72 predicted_time_scaled = svr_model.predict(predicted_distance_scaled)
73 predicted_time = scaler_y_svr.inverse_transform(predicted_time_scaled)
74
75 # Print the predicted travel time
76 print(f"Predicted travel time to next bus stop: {predicted_time[0]}")
77

```

```

-----
NameError                                Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_29236\2455553422.py in <module>
    47 # Reshape distances for SVR input
    48 X_svr = np.array(distances).reshape(-1, 1)
--> 49 y_svr = travel_times[1:]
    50
    51 # Standardize features for SVR

NameError: name 'travel_times' is not defined

```



In [19]:

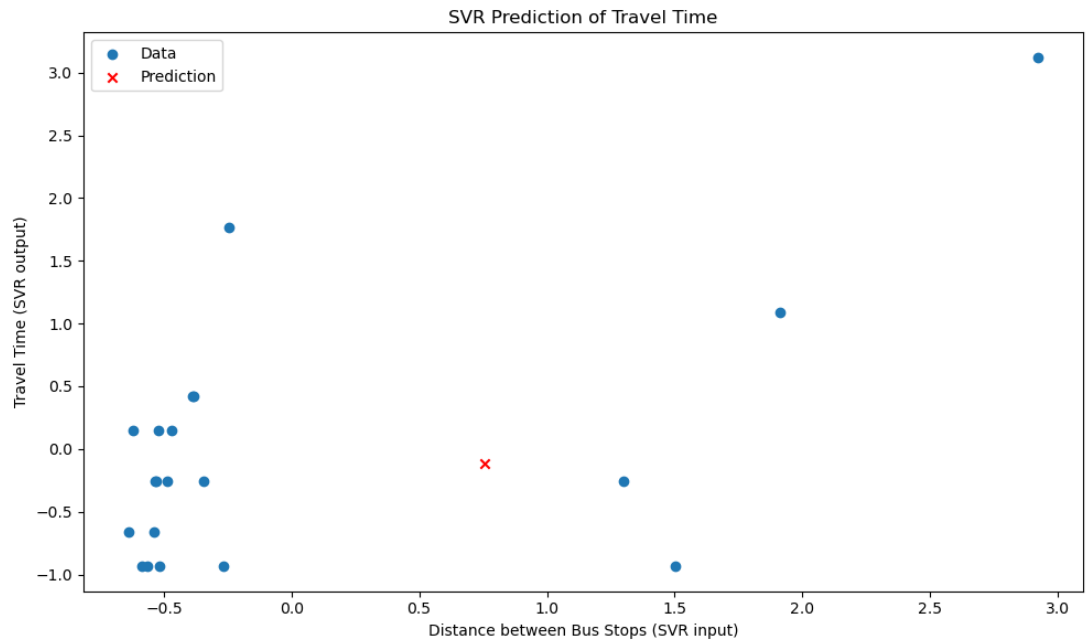
```
1 import numpy as np
2 from sklearn.neighbors import KNeighborsRegressor
3 from sklearn.svm import SVR
4 from sklearn.preprocessing import StandardScaler
5 import matplotlib.pyplot as plt
6 from geopy.distance import geodesic
7
8 # Example data: coordinates of bus stops and travel times (in minutes)
9 bus_stops = np.array([
10     [17.2404, 78.4293], # Shamshabad, Hyderabad
11     [17.5453, 78.3408], # Narkuda, Hyderabad
12     [17.1582, 78.1874], # Shabad, Hyderabad
13     [17.7275, 78.2746], # Rajiv Gandhi Airport
14     [17.4265, 78.4585], # Gandipet, Hyderabad
15     [17.4646, 78.4859], # Ocean Park, Hyderabad
16     [17.4125, 78.4388], # MehdiPatnam, Hyderabad
17     [17.3986, 78.4703], # Golconda Fort, Hyderabad
18     [17.3822, 78.4867], # Qutub Shahi Tombs, Hyderabad
19     [17.3850, 78.4804], # Toli Chowki, Hyderabad
20     [17.4090, 78.4896], # Banjara Hills, Hyderabad
21     [17.4349, 78.4484], # Film Nagar, Hyderabad
22     [17.4396, 78.4170], # HITEC City, Hyderabad
23     [17.4461, 78.3490], # JNTU, Hyderabad
24     [17.4485, 78.3719], # Kukatpally, Hyderabad
25     [17.4531, 78.3636], # Moosapet, Hyderabad
26     [17.4711, 78.3656], # Bharatnagar, Hyderabad
27     [17.4861, 78.3850], # Ameerpet, Hyderabad
28     [17.5099, 78.3765], # Punjagutta, Hyderabad
29     [17.5184, 78.3893], # Nagarjuna Circle, Hyderabad
30     [17.4811, 78.4291], # Masab Tank, Hyderabad
31 ])
32
33 bus_stop_names = [
34     "Shamshabad", "Narkuda", "Shabad", "Rajiv Gandhi Airport", "Gandipet",
35     "Ocean Park", "MehdiPatnam", "Golconda Fort", "Qutub Shahi Tomb",
36     "Banjara Hills", "Film Nagar", "HITEC City", "JNTU", "Kukatpally",
37     "Moosapet", "Bharatnagar", "Ameerpet", "Punjagutta", "Nagarjuna Circle",
38 ]
39
40 travel_times = np.array([0, 15, 25, 40, 10, 20, 30, 18, 15, 12, 10, 15, 10, 12, 10, 15, 10, 12, 10, 15])
41
42 # Calculate distances between bus stops using geopy's geodesic
43 distances = [geodesic(bus_stops[i], bus_stops[i+1]).kilometers for i in range(len(bus_stops)-1)]
44
45 # Reshape distances for KNN input
46 X_knn = np.array(distances).reshape(-1, 1)
47 y_knn = bus_stops[1:]
48
49 # Standardize features for KNN
50 scaler_X_knn = StandardScaler()
51 X_scaled_knn = scaler_X_knn.fit_transform(X_knn)
52
53 # Train KNN model
54 knn_model = KNeighborsRegressor(n_neighbors=3)
55 knn_model.fit(X_scaled_knn, y_knn)
56
57 # Reshape distances for SVR input
58 X_svr = np.array(distances).reshape(-1, 1)
59 y_svr = travel_times[1:]
60
61 # Standardize features for SVR
```

```

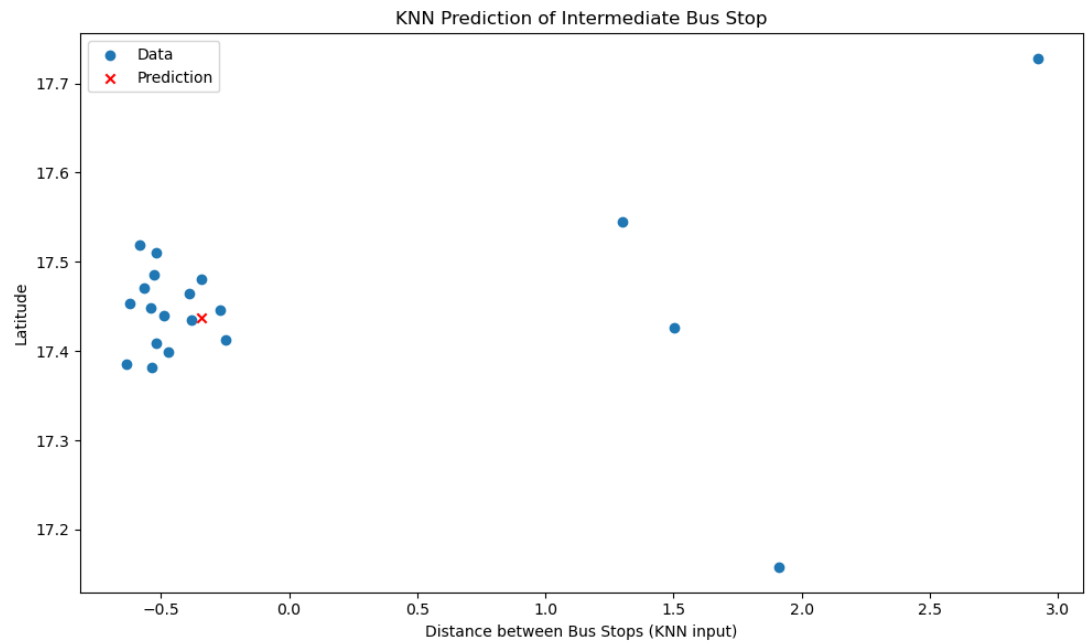
62 scaler_X_svr = StandardScaler()
63 scaler_y_svr = StandardScaler()
64 X_scaled_svr = scaler_X_svr.fit_transform(X_svr)
65 y_scaled_svr = scaler_y_svr.fit_transform(y_svr.reshape(-1, 1))
66
67 # Train SVR model
68 svr_model = SVR(kernel='rbf')
69 svr_model.fit(X_scaled_svr, y_scaled_svr.ravel())
70
71 # Custom input distance
72 source_stop_index = 0
73 destination_stop_index = 5
74 input_distance = geodesic(bus_stops[source_stop_index], bus_stops[destination_stop_index])
75
76 # Predict intermediate point using KNN
77 predicted_point = knn_model.predict([[input_distance]])
78 predicted_distance = np.array([geodesic(bus_stops[source_stop_index], predicted_point)])
79
80 # Scale predicted distance for SVR
81 predicted_distance_scaled = scaler_X_svr.transform(predicted_distance.reshape(-1, 1))
82
83 # Predict travel time using SVR
84 predicted_time_scaled = svr_model.predict(predicted_distance_scaled)
85 predicted_time = scaler_y_svr.inverse_transform(predicted_time_scaled)
86
87 # Print the predicted travel time
88 print(f"Predicted travel time from {bus_stop_names[source_stop_index]} to {bus_stop_names[destination_stop_index]}: {predicted_time[0]} minutes")
89
90
91 # Visualization: SVR prediction of travel time
92 plt.figure(figsize=(10, 6))
93 plt.scatter(X_scaled_svr, y_scaled_svr, label='Data')
94 plt.scatter(predicted_distance_scaled, predicted_time_scaled, color='red', label='Prediction')
95 plt.xlabel('Distance between Bus Stops (SVR input)')
96 plt.ylabel('Travel Time (SVR output)')
97 plt.title('SVR Prediction of Travel Time')
98 plt.legend()
99
100 plt.tight_layout()
101 plt.show()
102

```

Predicted travel time from Shamshabad to Ocean Park: 16.03 minutes

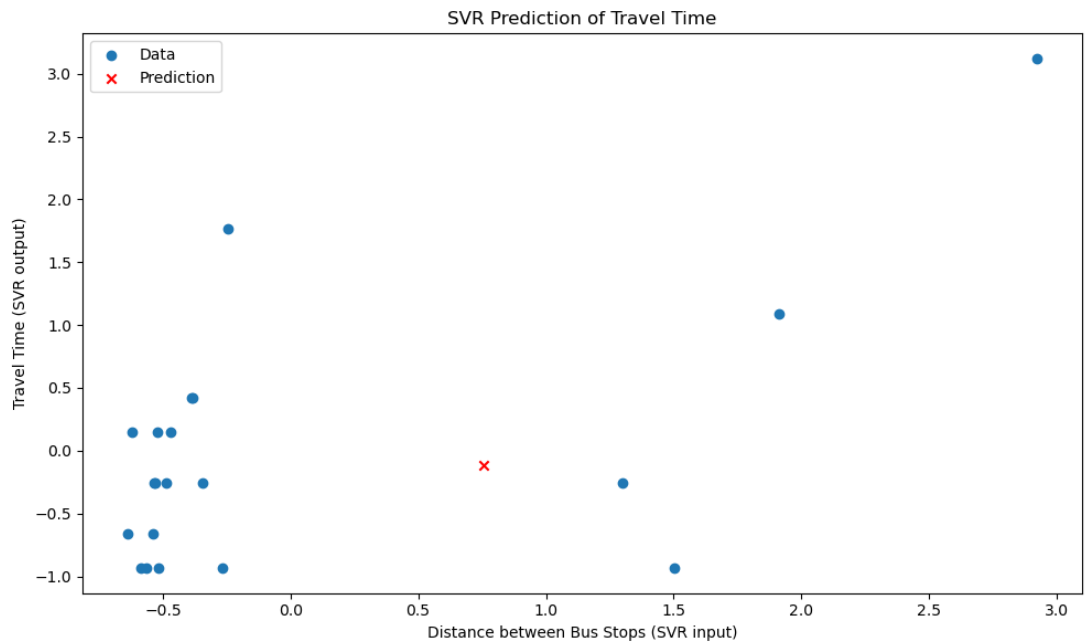


```
In [20]: ▶ 1 # Visualization: KNN prediction of intermediate point
2 plt.figure(figsize=(10, 6))
3 plt.scatter(X_scaled_knn, y_knn[:, 0], label='Data')
4 predicted_lat = knn_model.predict([[X_knn[-1][0]]])[0][0]
5 plt.scatter(X_scaled_knn[-1], predicted_lat, color='red', marker='x')
6 plt.xlabel('Distance between Bus Stops (KNN input)')
7 plt.ylabel('Latitude')
8 plt.title('KNN Prediction of Intermediate Bus Stop')
9 plt.legend()
10
11 plt.tight_layout()
12 plt.show()
```



In [21]: 

```
1 # Visualization: SVR prediction of travel time
2 plt.figure(figsize=(10, 6))
3 plt.scatter(X_scaled_svr, y_scaled_svr, label='Data')
4 plt.scatter(predicted_distance_scaled, predicted_time_scaled, color
5 plt.xlabel('Distance between Bus Stops (SVR input)')
6 plt.ylabel('Travel Time (SVR output)')
7 plt.title('SVR Prediction of Travel Time')
8 plt.legend()
9
10 plt.tight_layout()
11 plt.show()
```

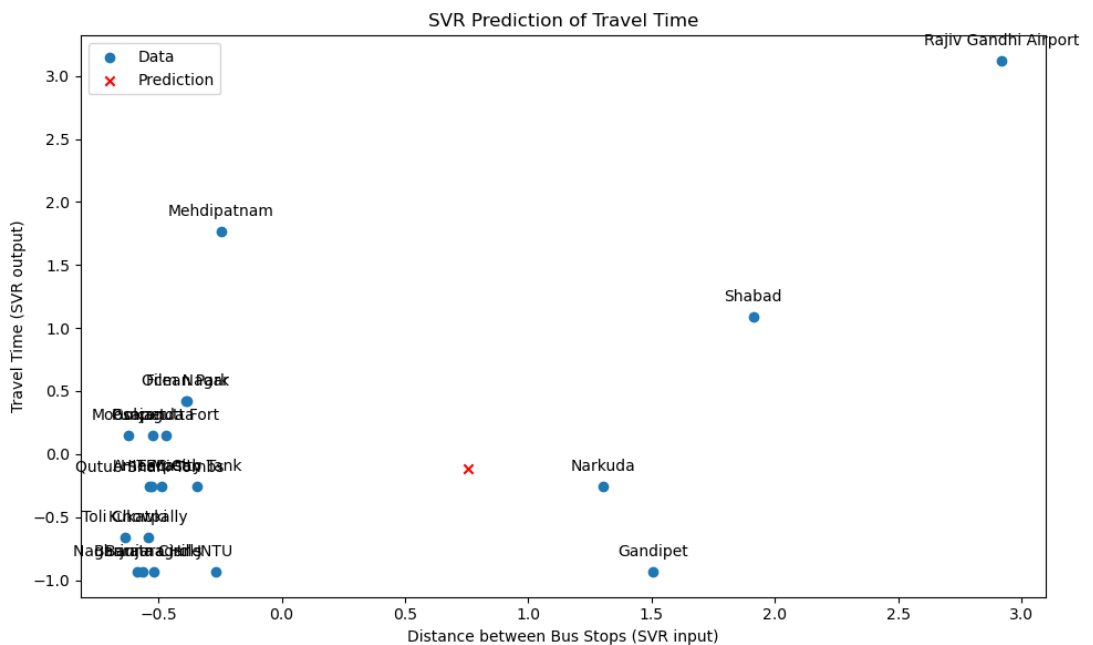


In [22]: 

```

1  # Visualization: SVR prediction of travel time
2  plt.figure(figsize=(10, 6))
3  plt.scatter(X_scaled_svr, y_scaled_svr, label='Data')
4
5  # Add data labels for each point
6  for i, txt in enumerate(bus_stop_names[1:]):
7      plt.annotate(txt, (X_scaled_svr[i], y_scaled_svr[i]), textcoord
8
9  # Add a red 'x' marker for the predicted point
10 plt.scatter(predicted_distance_scaled, predicted_time_scaled, color
11
12 plt.xlabel('Distance between Bus Stops (SVR input)')
13 plt.ylabel('Travel Time (SVR output)')
14 plt.title('SVR Prediction of Travel Time')
15 plt.legend()
16
17 plt.tight_layout()
18 plt.show()
19

```

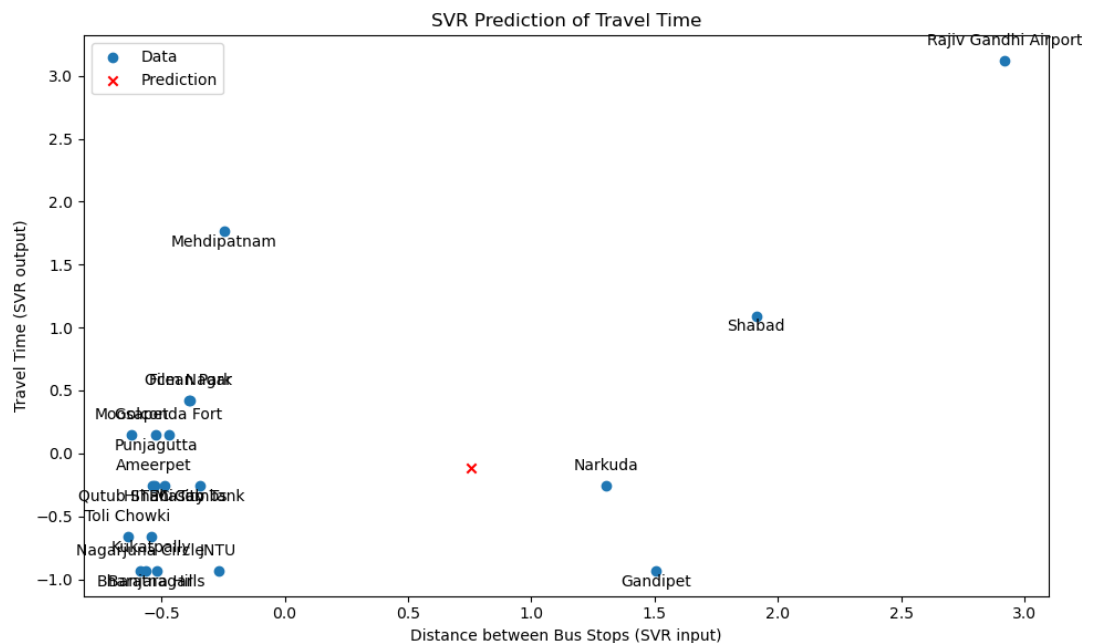


In [23]:

```

1 # Visualization: SVR prediction of travel time
2 plt.figure(figsize=(10, 6))
3 plt.scatter(X_scaled_svr, y_scaled_svr, label='Data')
4
5 # Add data labels for each point with adjusted positioning
6 for i, txt in enumerate(bus_stop_names[1:]):
7     if i % 2 == 0:
8         plt.annotate(txt, (X_scaled_svr[i], y_scaled_svr[i]), textc
9     else:
10        plt.annotate(txt, (X_scaled_svr[i], y_scaled_svr[i]), textc
11
12 # Add a red 'x' marker for the predicted point
13 plt.scatter(predicted_distance_scaled, predicted_time_scaled, color
14
15 plt.xlabel('Distance between Bus Stops (SVR input)')
16 plt.ylabel('Travel Time (SVR output)')
17 plt.title('SVR Prediction of Travel Time')
18 plt.legend()
19
20 plt.tight_layout()
21 plt.show()
22

```

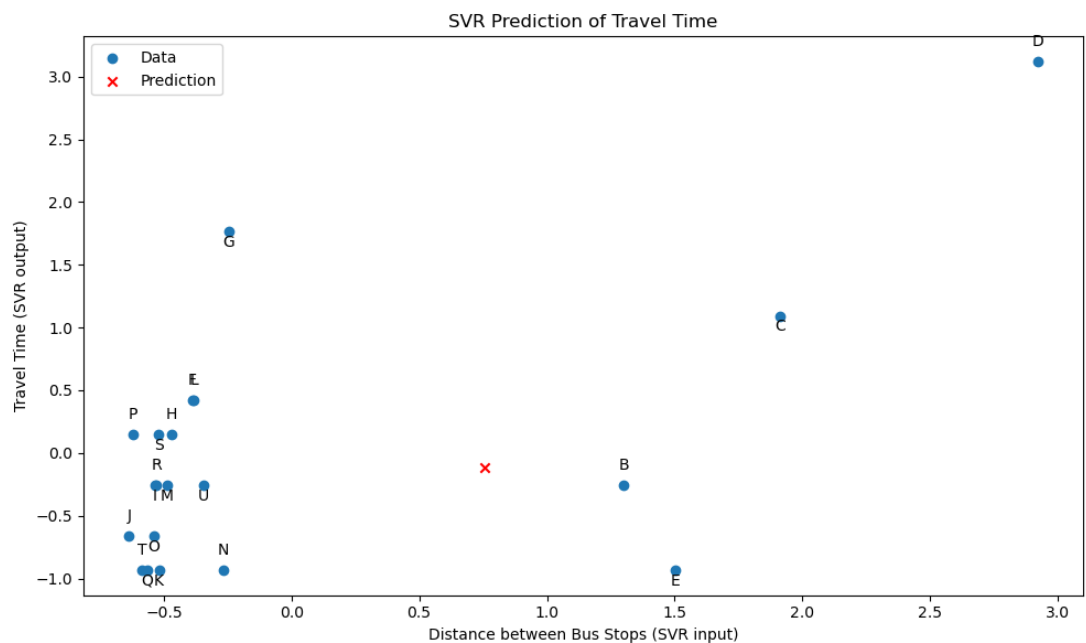




```

In [24]: 1 # Short names for bus stops
2 bus_stop_short_names = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
3
4 # Visualization: SVR prediction of travel time
5 plt.figure(figsize=(10, 6))
6 plt.scatter(X_scaled_svr, y_scaled_svr, label='Data')
7
8 # Add data labels for each point with short names
9 for i, txt in enumerate(bus_stop_short_names[1:]):
10     if i % 2 == 0:
11         plt.annotate(txt, (X_scaled_svr[i], y_scaled_svr[i]), textc
12     else:
13         plt.annotate(txt, (X_scaled_svr[i], y_scaled_svr[i]), textc
14
15 # Add a red 'x' marker for the predicted point
16 plt.scatter(predicted_distance_scaled, predicted_time_scaled, color
17
18 plt.xlabel('Distance between Bus Stops (SVR input)')
19 plt.ylabel('Travel Time (SVR output)')
20 plt.title('SVR Prediction of Travel Time')
21 plt.legend()
22
23 plt.tight_layout()
24 plt.show()
25

```



```
In [1]: ▶ 1 import pygame
          2
          3 # Initialize pygame
          4 pygame.init()
          5
          6 # Load a sound file
          7 sound_file = "mixkit-game-show-suspense-waiting-667.wav"
          8 pygame.mixer.music.load(sound_file)
          9
         10 # Play the sound
         11 pygame.mixer.music.play()
         12
         13 # Wait for the sound to finish playing
         14 while pygame.mixer.music.get_busy():
         15     pygame.time.Clock().tick(10)
         16
         17 # Quit pygame
         18 pygame.quit()
         19
```

pygame 2.5.1 (SDL 2.28.2, Python 3.9.13)

Hello from the pygame community. <https://www.pygame.org/contribute.html>

1 (<https://www.pygame.org/contribute.html>)



In [27]:

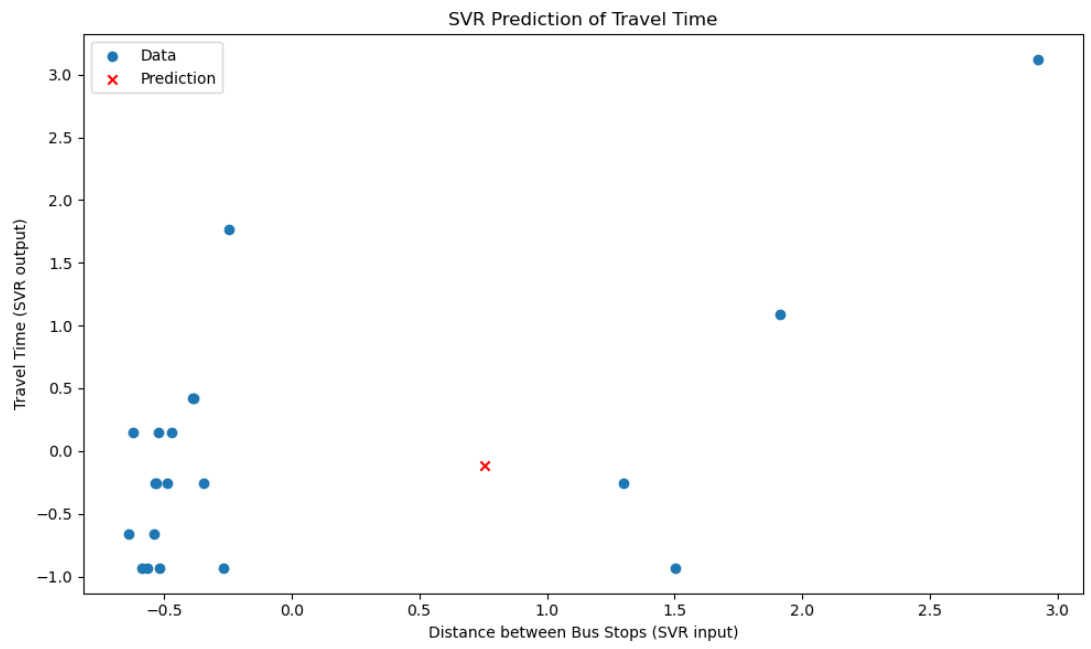
```
1 import numpy as np
2 from sklearn.neighbors import KNeighborsRegressor
3 from sklearn.svm import SVR
4 from sklearn.preprocessing import StandardScaler
5 import matplotlib.pyplot as plt
6 from geopy.distance import geodesic
7
8 # Example data: coordinates of bus stops and travel times (in minutes)
9 bus_stops = np.array([
10     [17.2404, 78.4293], # Shamshabad, Hyderabad
11     [17.5453, 78.3408], # Narkuda, Hyderabad
12     [17.1582, 78.1874], # Shabad, Hyderabad
13     [17.7275, 78.2746], # Rajiv Gandhi Airport
14     [17.4265, 78.4585], # Gandipet, Hyderabad
15     [17.4646, 78.4859], # Ocean Park, Hyderabad
16     [17.4125, 78.4388], # Mehdiapatnam, Hyderabad
17     [17.3986, 78.4703], # Golconda Fort, Hyderabad
18     [17.3822, 78.4867], # Qutub Shahi Tombs, Hyderabad
19     [17.3850, 78.4804], # Toli Chowki, Hyderabad
20     [17.4090, 78.4896], # Banjara Hills, Hyderabad
21     [17.4349, 78.4484], # Film Nagar, Hyderabad
22     [17.4396, 78.4170], # HITEC City, Hyderabad
23     [17.4461, 78.3490], # JNTU, Hyderabad
24     [17.4485, 78.3719], # Kukatpally, Hyderabad
25     [17.4531, 78.3636], # Moosapet, Hyderabad
26     [17.4711, 78.3656], # Bharatnagar, Hyderabad
27     [17.4861, 78.3850], # Ameerpet, Hyderabad
28     [17.5099, 78.3765], # Punjagutta, Hyderabad
29     [17.5184, 78.3893], # Nagarjuna Circle, Hyderabad
30     [17.4811, 78.4291], # Masab Tank, Hyderabad
31 ])
32
33 bus_stop_names = [
34     "Shamshabad", "Narkuda", "Shabad", "Rajiv Gandhi Airport", "Gandipet",
35     "Ocean Park", "Mehdiapatnam", "Golconda Fort", "Qutub Shahi Tomb",
36     "Banjara Hills", "Film Nagar", "HITEC City", "JNTU", "Kukatpally",
37     "Moosapet", "Bharatnagar", "Ameerpet", "Punjagutta", "Nagarjuna Circle",
38 ]
39
40 travel_times = np.array([0, 15, 25, 40, 10, 20, 30, 18, 15, 12, 10, 15, 10, 12, 10, 15, 10, 12, 10, 15])
41
42 # Calculate distances between bus stops using geopy's geodesic
43 distances = [geodesic(bus_stops[i], bus_stops[i+1]).kilometers for i in range(len(bus_stops)-1)]
44
45 # Reshape distances for KNN input
46 X_knn = np.array(distances).reshape(-1, 1)
47 y_knn = bus_stops[1:]
48
49 # Standardize features for KNN
50 scaler_X_knn = StandardScaler()
51 X_scaled_knn = scaler_X_knn.fit_transform(X_knn)
52
53 # Train KNN model
54 knn_model = KNeighborsRegressor(n_neighbors=3)
55 knn_model.fit(X_scaled_knn, y_knn)
56
57 # Reshape distances for SVR input
58 X_svr = np.array(distances).reshape(-1, 1)
59 y_svr = travel_times[1:]
60
61 # Standardize features for SVR
```

```

62 scaler_X_svr = StandardScaler()
63 scaler_y_svr = StandardScaler()
64 X_scaled_svr = scaler_X_svr.fit_transform(X_svr)
65 y_scaled_svr = scaler_y_svr.fit_transform(y_svr.reshape(-1, 1))
66
67 # Train SVR model
68 svr_model = SVR(kernel='rbf')
69 svr_model.fit(X_scaled_svr, y_scaled_svr.ravel())
70
71 # Custom input distance
72 source_stop_index = 0
73 destination_stop_index = 5
74 input_distance = geodesic(bus_stops[source_stop_index], bus_stops[destination_stop_index])
75
76 # Predict intermediate point using KNN
77 predicted_point = knn_model.predict([[input_distance]])
78 predicted_distance = np.array([geodesic(bus_stops[source_stop_index], predicted_point)])
79
80 # Scale predicted distance for SVR
81 predicted_distance_scaled = scaler_X_svr.transform(predicted_distance.reshape(-1, 1))
82
83 # Predict travel time using SVR
84 predicted_time_scaled = svr_model.predict(predicted_distance_scaled)
85 predicted_time = scaler_y_svr.inverse_transform(predicted_time_scaled)
86
87 # Print the predicted travel time
88 print(f"Predicted travel time from {bus_stop_names[source_stop_index]} to {bus_stop_names[destination_stop_index]}: {predicted_time[0]} minutes")
89
90
91 # Visualization: SVR prediction of travel time
92 plt.figure(figsize=(10, 6))
93 plt.scatter(X_scaled_svr, y_scaled_svr, label='Data')
94 plt.scatter(predicted_distance_scaled, predicted_time_scaled, color='red', label='Prediction')
95 plt.xlabel('Distance between Bus Stops (SVR input)')
96 plt.ylabel('Travel Time (SVR output)')
97 plt.title('SVR Prediction of Travel Time')
98 plt.legend()
99
100 plt.tight_layout()
101 plt.show()
102

```

Predicted travel time from Shamshabad to Ocean Park: 16.03 minutes





In [29]:

```
1 import numpy as np
2 from sklearn.svm import SVR
3 from sklearn.preprocessing import StandardScaler
4 import matplotlib.pyplot as plt
5 from geopy.distance import geodesic
6 import time
7 import pygame
8
9 # Example data: coordinates of bus stops and travel times (in minutes)
10 bus_stops = np.array([
11     [17.2404, 78.4293], # Shamshabad, Hyderabad
12     [17.5453, 78.3408], # Narkuda, Hyderabad
13     [17.1582, 78.1874], # Shabad, Hyderabad
14     [17.7275, 78.2746], # Rajiv Gandhi International Airport (not
15     [17.4265, 78.4585], # Gandipet, Hyderabad
16     [17.4646, 78.4859], # Ocean Park, Hyderabad
17     [17.4125, 78.4388], # MehdiPatnam, Hyderabad
18     [17.3986, 78.4703], # Golconda Fort, Hyderabad
19     [17.3822, 78.4867], # Qutub Shahi Tombs, Hyderabad
20     [17.3850, 78.4804], # Toli Chowki, Hyderabad
21     [17.4090, 78.4896], # Banjara Hills, Hyderabad
22     [17.4349, 78.4484], # Film Nagar, Hyderabad
23     [17.4396, 78.4170], # HITEC City, Hyderabad
24     [17.4461, 78.3490], # JNTU, Hyderabad
25     [17.4485, 78.3719], # Kukatpally, Hyderabad
26     [17.4531, 78.3636], # Moosapet, Hyderabad
27     [17.4711, 78.3656], # Bharatnagar, Hyderabad
28     [17.4861, 78.3850], # Ameerpet, Hyderabad
29     [17.5099, 78.3765], # Punjagutta, Hyderabad
30     [17.5184, 78.3893], # Nagarjuna Circle, Hyderabad
31     [17.4811, 78.4291], # Masab Tank, Hyderabad
32 ])
33
34 bus_stop_names = [
35     "Shamshabad", "Narkuda", "Shabad", "Rajiv Gandhi Airport", "Gandipet",
36     "Ocean Park", "MehdiPatnam", "Golconda Fort", "Qutub Shahi Tombs",
37     "Banjara Hills", "Film Nagar", "HITEC City", "JNTU", "Kukatpally",
38     "Moosapet", "Bharatnagar", "Ameerpet", "Punjagutta", "Nagarjuna Circle",
39 ]
40
41 travel_times = np.array([0, 15, 25, 40, 10, 20, 30, 18, 15, 12, 10, 15, 10, 12, 10, 15, 10, 12, 10, 15, 10])
42
43 # Calculate distances between bus stops using geopy's geodesic
44 distances = [geodesic(bus_stops[i], bus_stops[i+1]).kilometers for i in range(len(bus_stops)-1)]
45
46 # Reshape distances and travel times for SVR input
47 X = np.array(distances).reshape(-1, 1)
48 y = travel_times[1:]
49
50 # Standardize features
51 scaler_X = StandardScaler()
52 scaler_y = StandardScaler()
53 X_scaled = scaler_X.fit_transform(X)
54 y_scaled = scaler_y.fit_transform(y.reshape(-1, 1))
55
56 # Train SVR model
57 svr_model = SVR(kernel='rbf')
58 svr_model.fit(X_scaled, y_scaled.ravel()) # Ravel y_scaled to make it 1D
59
60 # Custom input distance
61 input_distance = geodesic(bus_stops[0], bus_stops[1]).kilometers
```



```

62
63 # Scale input distance
64 input_distance_scaled = scaler_X.transform(np.array([[input_distance
65
66 # Predict travel time using SVR
67 predicted_time_scaled = svr_model.predict(input_distance_scaled)
68 predicted_time = scaler_y.inverse_transform(predicted_time_scaled.r
69
70 # Print the predicted travel time
71 print(f"Predicted travel time to next bus stop: {predicted_time[0][
72
73 # Print the predicted travel time
74 source_stop_name = bus_stop_names[source_stop_index]
75 destination_stop_name = bus_stop_names[destination_stop_index]
76 print(f"Predicted travel time from {source_stop_name} to {destinati
77
78 # Play an alarm sound 2 minutes before reaching the predicted trave
79 alarm_time = predicted_time[0][0] - 2 # 2 minutes before predicted
80 current_time = time.time()
81 time_to_wait = alarm_time * 60 - current_time # Convert to seconds
82
83 # Play the alarm sound 2 minutes before reaching the destination
84 if time_to_wait > 0:
85     time.sleep(time_to_wait)
86
87     # Initialize Pygame mixer
88     pygame.mixer.init()
89
90     # Load and play the alarm sound
91     pygame.mixer.music.load("mixkit-game-show-suspense-waiting-667.
92     pygame.mixer.music.play()
93
94     # Wait for the sound to finish playing
95     while pygame.mixer.music.get_busy():
96         pygame.time.Clock().tick(10)
97
98 # Visualization: SVR prediction
99 plt.figure(figsize=(10, 6))
100 plt.scatter(X_scaled, y_scaled, label='Data')
101 plt.scatter(input_distance_scaled, predicted_time_scaled, color='re
102 plt.xlabel('Distance between Bus Stops')
103 plt.ylabel('Travel Time (minutes)')
104 plt.title('SVR Prediction of Travel Time')
105 plt.legend()
106
107 # Annotate bus stop names on the graph
108 for i, name in enumerate(bus_stop_names[1:]):
109     plt.annotate(name, (X_scaled[i][0], y_scaled[i][0]), textcoords
110
111 plt.tight_layout()
112 plt.show()
113
114 # Visualization: Original data points
115 plt.figure(figsize=(10, 6))
116 plt.scatter(X, y)
117 plt.xlabel('Distance between Bus Stops')
118 plt.ylabel('Travel Time (minutes)')
119 plt.title('Original Data Points')
120 plt.show()
121
122 # Visualization: SVR prediction with bus stop names

```

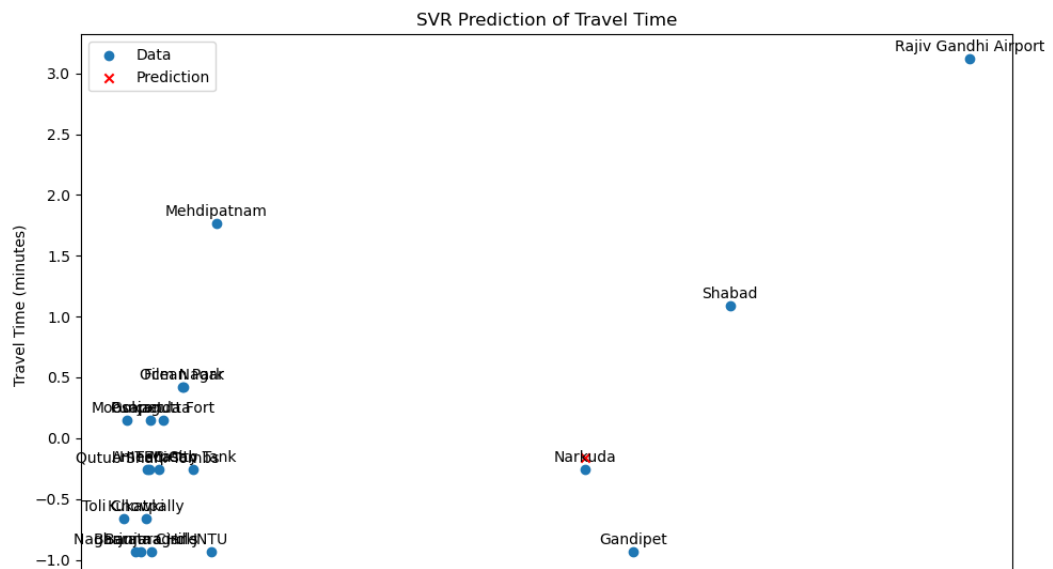
```

123 plt.figure(figsize=(10, 6))
124 plt.scatter(X_scaled, y_scaled, label='Data')
125 plt.scatter(input_distance_scaled, predicted_time_scaled, color='red')
126 plt.xlabel('Distance between Bus Stops')
127 plt.ylabel('Travel Time (minutes)')
128 plt.title('SVR Prediction of Travel Time')
129 plt.legend()
130
131 # Annotate bus stop names on the graph
132 for i, name in enumerate(bus_stop_names[1:]):
133     plt.annotate(name, (X_scaled[i][0], y_scaled[i][0]), textcoords='axesfraction')
134
135 plt.tight_layout()
136 plt.show()
137
138 # Visualization: Predicted travel time vs. Actual travel time
139 plt.figure(figsize=(10, 6))
140 plt.plot(y_scaled, label='Actual Travel Time')
141 plt.plot(predicted_time_scaled, label='Predicted Travel Time')
142 plt.xlabel('Bus Stop')
143 plt.ylabel('Normalized Travel Time')
144 plt.title('Predicted Travel Time vs. Actual Travel Time')
145 plt.legend()
146 plt.xticks(range(len(bus_stop_names[1:])), bus_stop_names[1:], rotation=45)
147 plt.tight_layout()
148 plt.show()
149

```

Predicted travel time to next bus stop: 15.74 minutes

Predicted travel time from Shamshabad to Ocean Park: 15.74 minutes



In [ ]: 1 '''Above one is complete code'''

In [ ]: 1

In [ ]: 1

In [ ]: 1

In [ ]: 1

In [ ]: 1

In [ ]: 1

In [ ]: 1

In [ ]: 1