Waiter Tips Prediction

In [2]:

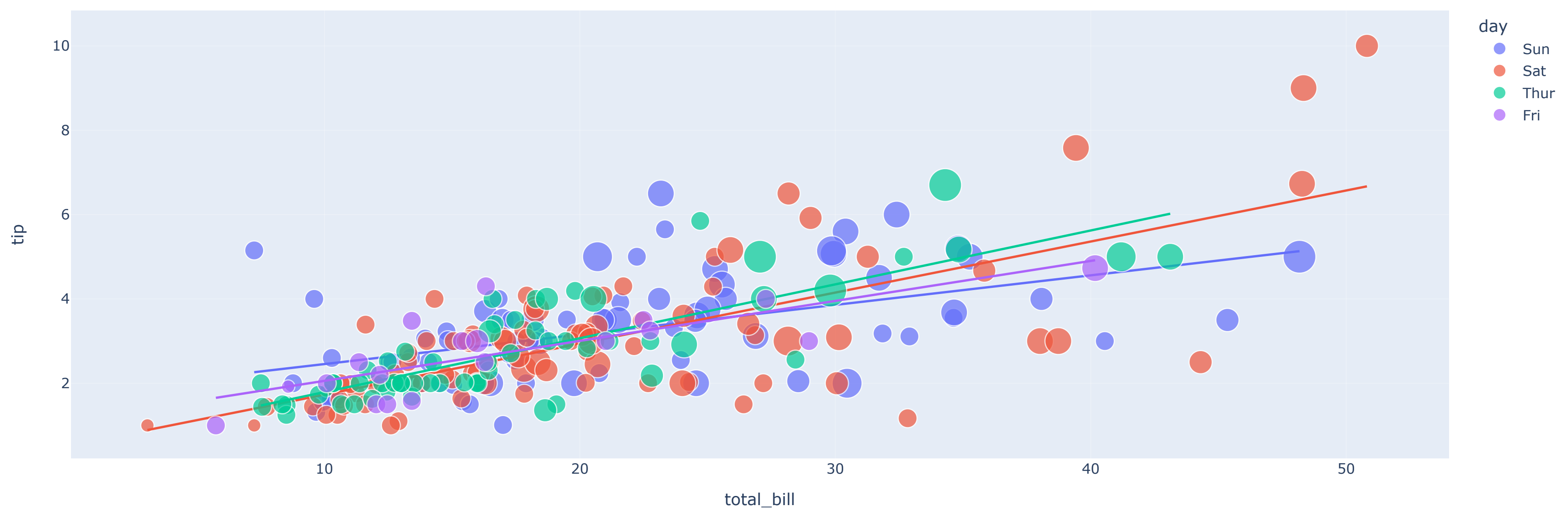
```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go

data = pd.read_csv("F:/udemy/waiterTip/tips.csv")
print(data.head())
```

```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

In [3]:

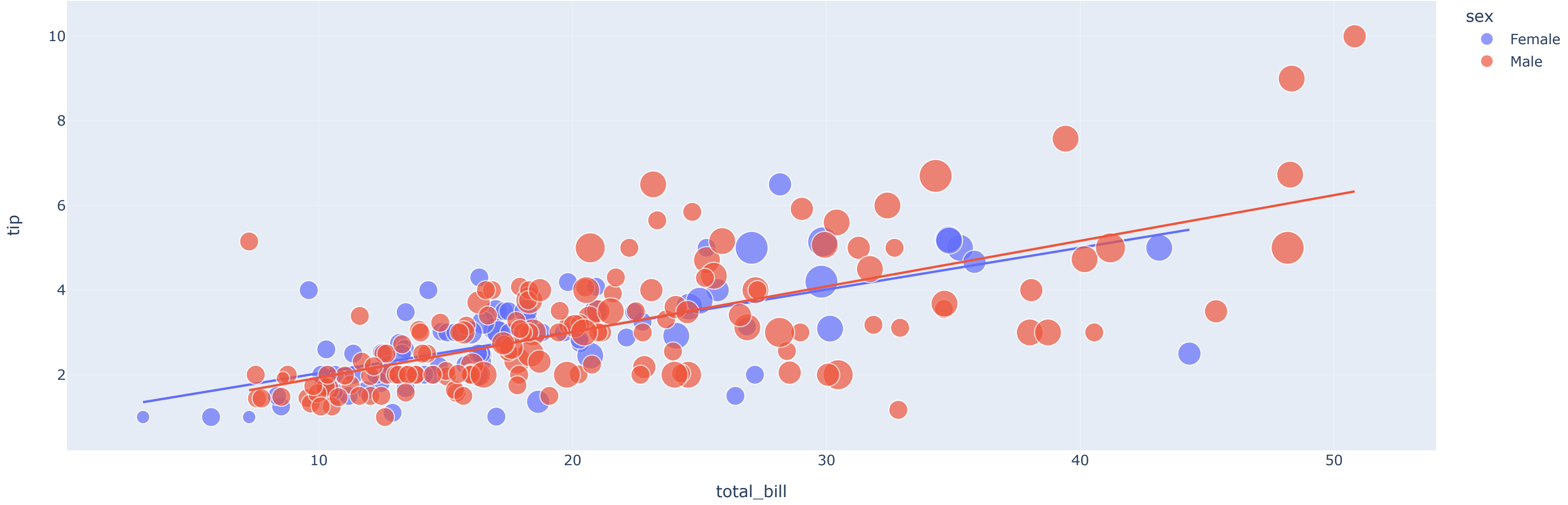
```
figure = px.scatter(data_frame = data, x="total_bill",
                    y="tip", size="size", color= "day", trendline="ols")
figure.show()
```



In the above graph we can see that the most tip was given on saturday.

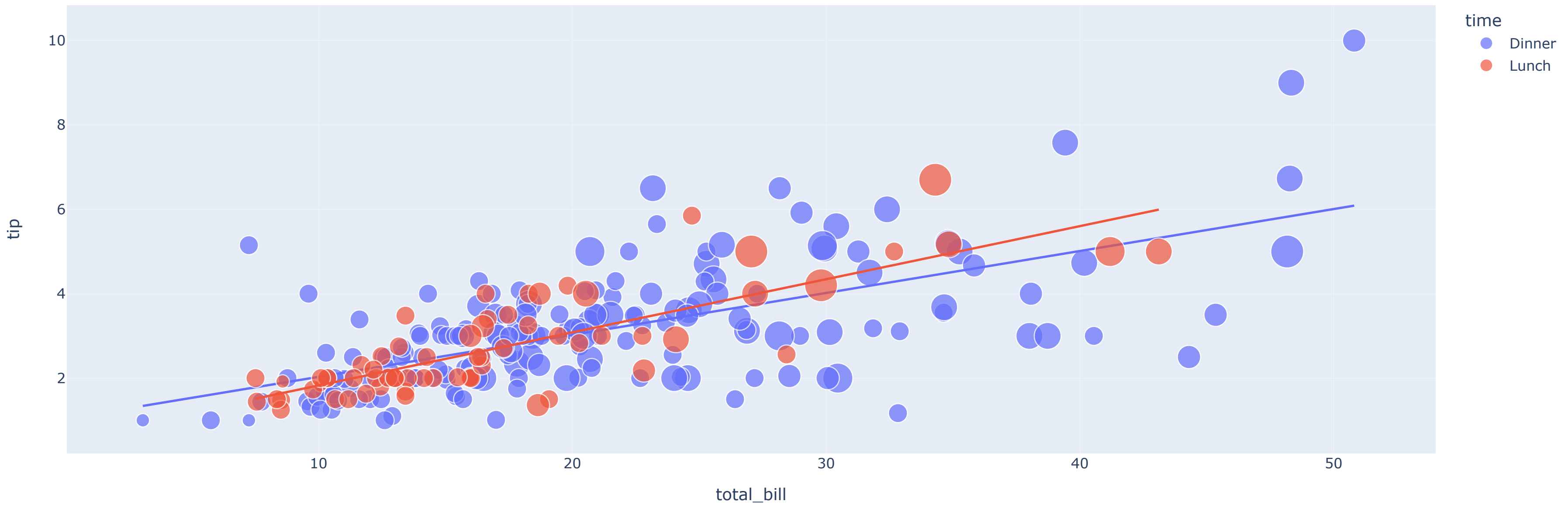
In [4]:

```
figure = px.scatter(data_frame = data, x="total_bill",
                    y="tip", size="size", color= "sex", trendline="ols")
figure.show()
```



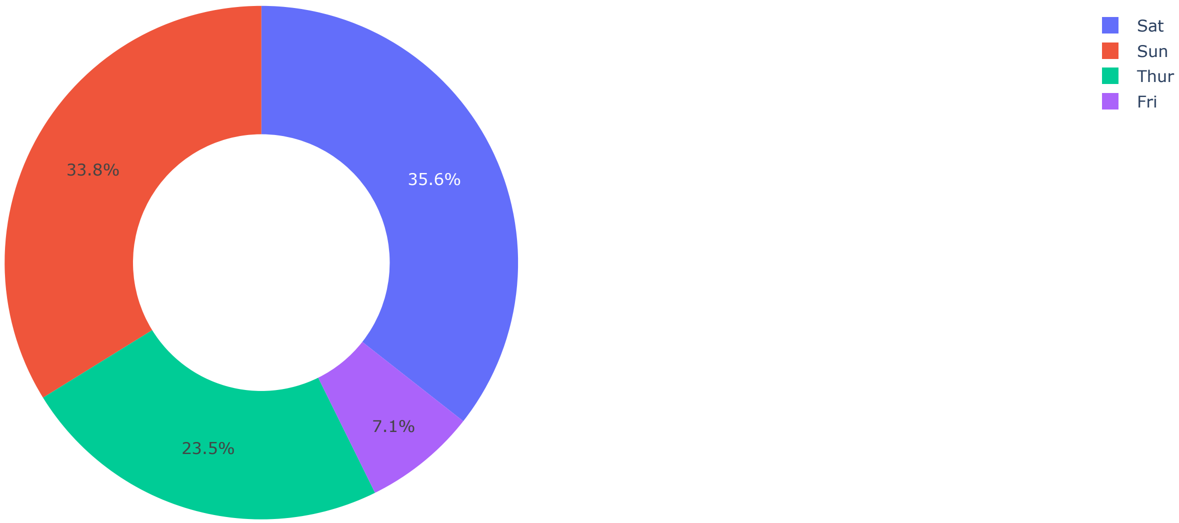
In the above grapf we can see that the most tip was given by male customers.

```
In [5]: figure = px.scatter(data_frame = data, x="total_bill",
                            y="tip", size="size", color= "time", trendline="ols")
        figure.show()
```



from this graph we can analyze that the most tip was given at dinner time.

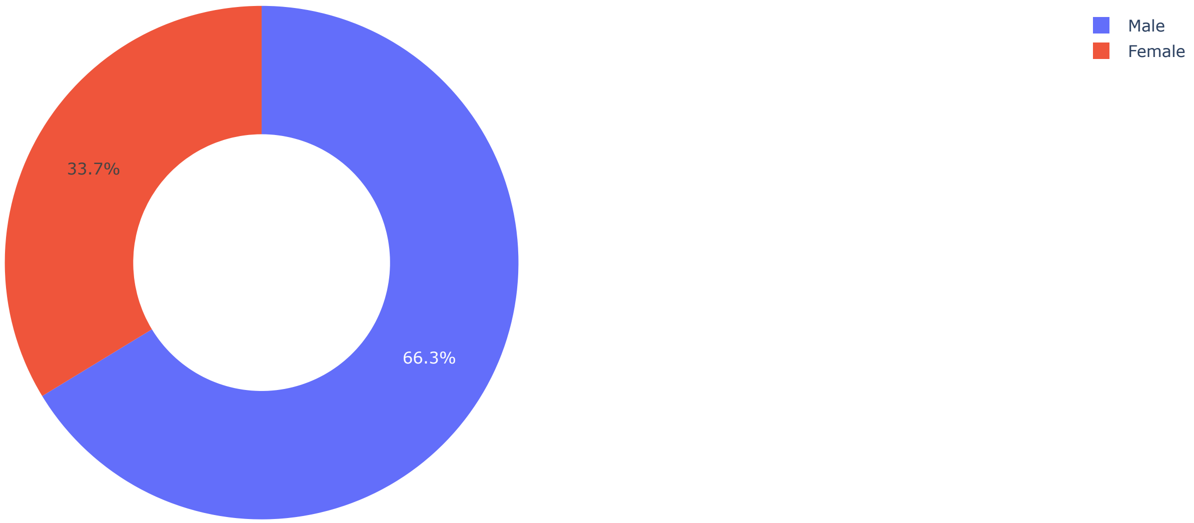
In [10]:
```
figure = px.pie(data,
             values='tip',
             names='day',hole = 0.5)
figure.show()
```



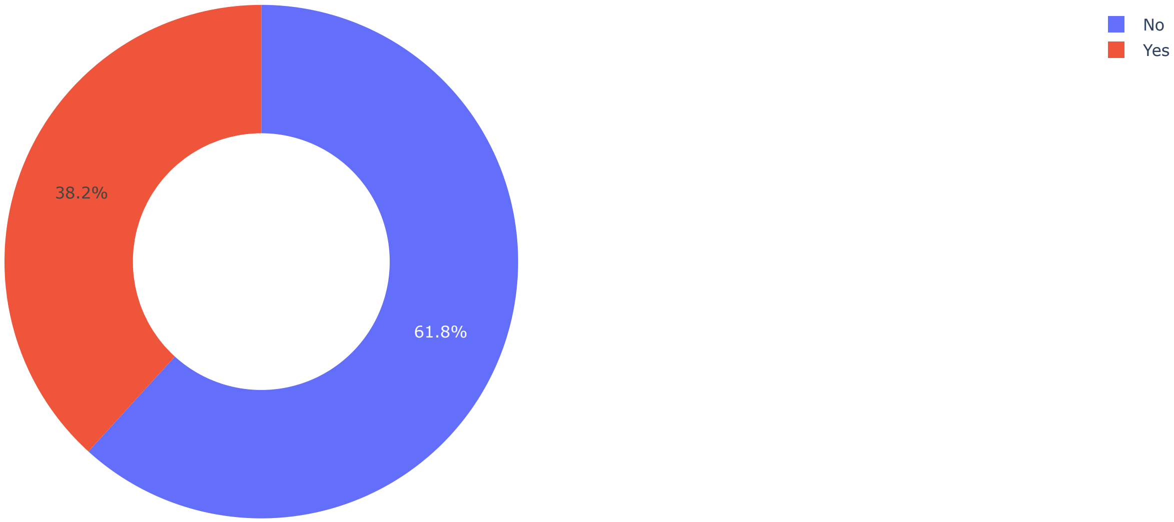
This chart express that the maximum tip was given on saturday and then sunday.

In [11]:

```
figure = px.pie(data,
             values='tip',
             names='sex',hole = 0.5)
figure.show()
```

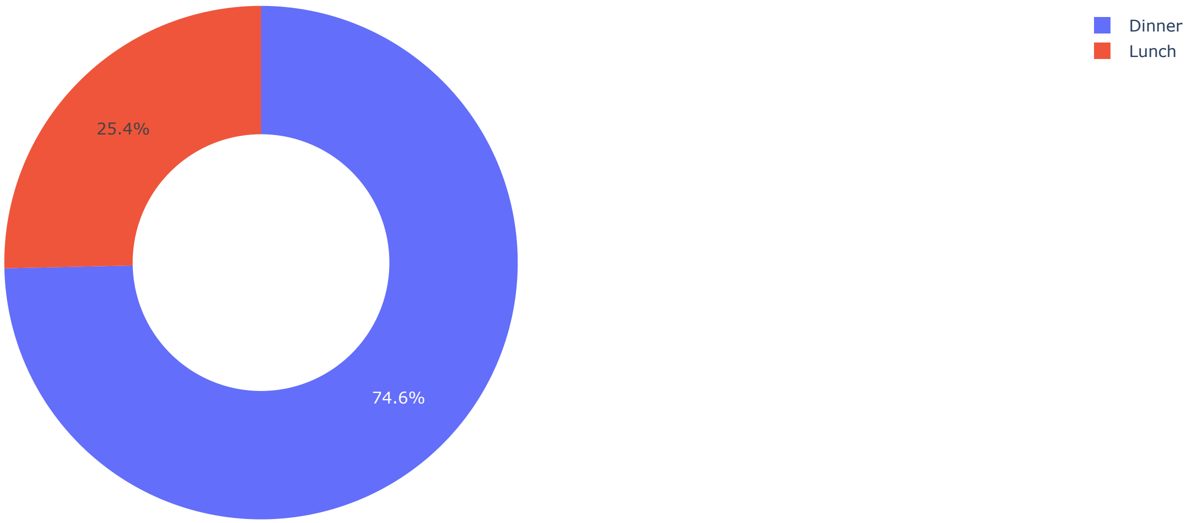


```
In [12]: figure = px.pie(data,
                      values='tip',
                      names='smoker',hole = 0.5)
         figure.show()
```



This chart shows that the non smoker customer gave more tips then the smoker customers.

```
In [13]: figure = px.pie(data,
                      values='tip',
                      names='time',hole = 0.5)
         figure.show()
```



Waiter Tips Prediction Model

Label Encoding

In [14]:

```
data["sex"] = data["sex"].map({"Female": 0, "Male": 1})
data["smoker"] = data["smoker"].map({"No": 0, "Yes": 1})
data["day"] = data["day"].map({"Thur": 0, "Fri": 1, "Sat": 2, "Sun": 3})
data["time"] = data["time"].map({"Lunch": 0, "Dinner": 1})
data.head()
```

Out[14]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	0	0	3	1	2
1	10.34	1.66	1	0	3	1	3
2	21.01	3.50	1	0	3	1	3
3	23.68	3.31	1	0	3	1	2
4	24.59	3.61	0	0	3	1	4

In [15]:

```
x = np.array(data[["total_bill", "sex", "smoker", "day",
                   "time", "size"]])
y = np.array(data["tip"])

from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.2,
                                                random_state=42)
```

In [16]:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(xtrain, ytrain)
```

Out[16]:

```
LinearRegression()
```

In [23]:

```
predicted = model.predict(xtest)
```

In [26]:

```
df = pd.DataFrame({'Actual': ytest,
     'Predicted': predicted})
```

In [28]:

```
print(df.head())
```

```
   Actual  Predicted
0    3.18   2.959150
1    2.00   1.979385
2    2.00   3.933555
3    5.16   3.815128
4    2.00   2.174782
```

In [30]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

mae = mean_absolute_error(ytest, predicted)
mse = mean_squared_error(ytest, predicted)
rmse = np.sqrt(mse)
r2 = r2_score(ytest, predicted)
```

Performance of the Model

In [31]:

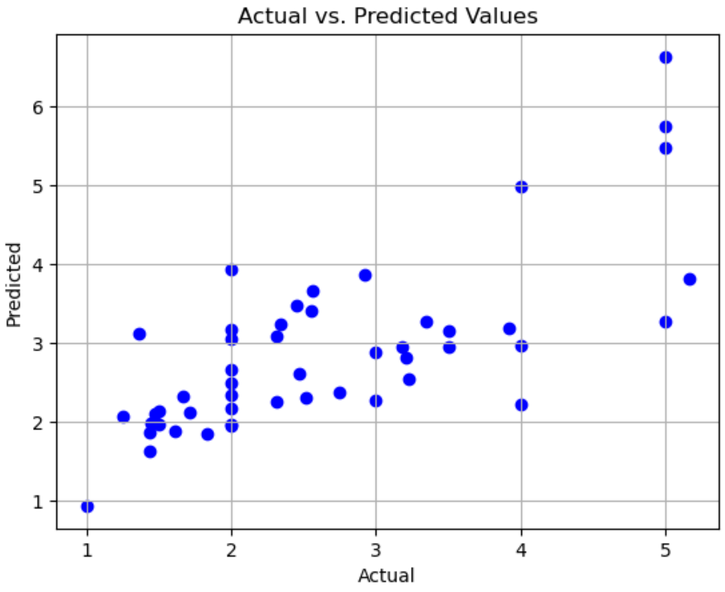
```
print('Mean Absolute Error (MAE) :', mae)
print('Mean Squared Error (MSE) :', mse)
print('Root Mean Absolute Error (RMSE) :', rmse)
print('R-squared (R²) Error (R²) :', r2)
```

```
Mean Absolute Error (MAE) : 0.6685728160722872
Mean Squared Error (MSE) : 0.6963090766605348
Root Mean Absolute Error (RMSE) : 0.8344513626692299
R-squared (R²) Error (R²) : 0.4429399687489899
```

In [37]:

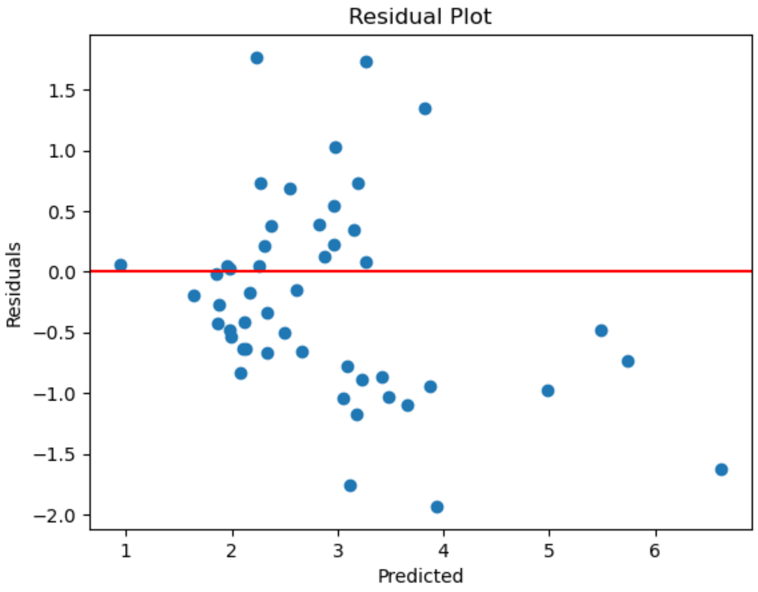
```
import matplotlib.pyplot as plt

plt.scatter(ytest, predicted, color='blue', label='Actual')
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("Actual vs. Predicted Values")
# Add grid lines to both x and y axes
plt.grid(True)
plt.show()
```



In [33]:

```
residuals = ytest - predicted
plt.scatter(predicted, residuals)
plt.xlabel("Predicted")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.axhline(y=0, color='r', linestyle='-')
plt.show()
```


Residual Plot
Residuals
Predicted