In [ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
```

In [ ]:
```
from google.colab import files
uploaded = files.upload()
df = pd.read_excel("synthetic_health_data.xlsx")
print(df)
```

Browse...

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving synthetic_health_data.xlsx to synthetic_health_data (2).xlsx
     Weight  Height  Age  Gender
0        88    1.70   43       0
1        78    1.69   64       0
2        64    1.57   86       0
3        92    1.67   64       0
4        57    1.66   35       1
..      ...     ...  ...     ...
495      54    1.67   36       1
496      61    1.87   33       0
497      65    1.64   79       0
498      75    1.73   53       1
499      75    1.75   47       0

[500 rows x 4 columns]
```

In [ ]:
```
df.shape
```

Out[ ]:
```
(500, 5)
```

In [ ]:
```
df['BMI'] = df['Weight'] / (df['Height'] ** 2)
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:

	0
Weight	0
Height	0
Age	0
Gender	0
BMI	0

dtype: int64

In [ ]:
```
x = df.drop("Weight", axis=1).values
y = df["Weight"].values
```

In [ ]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=
```

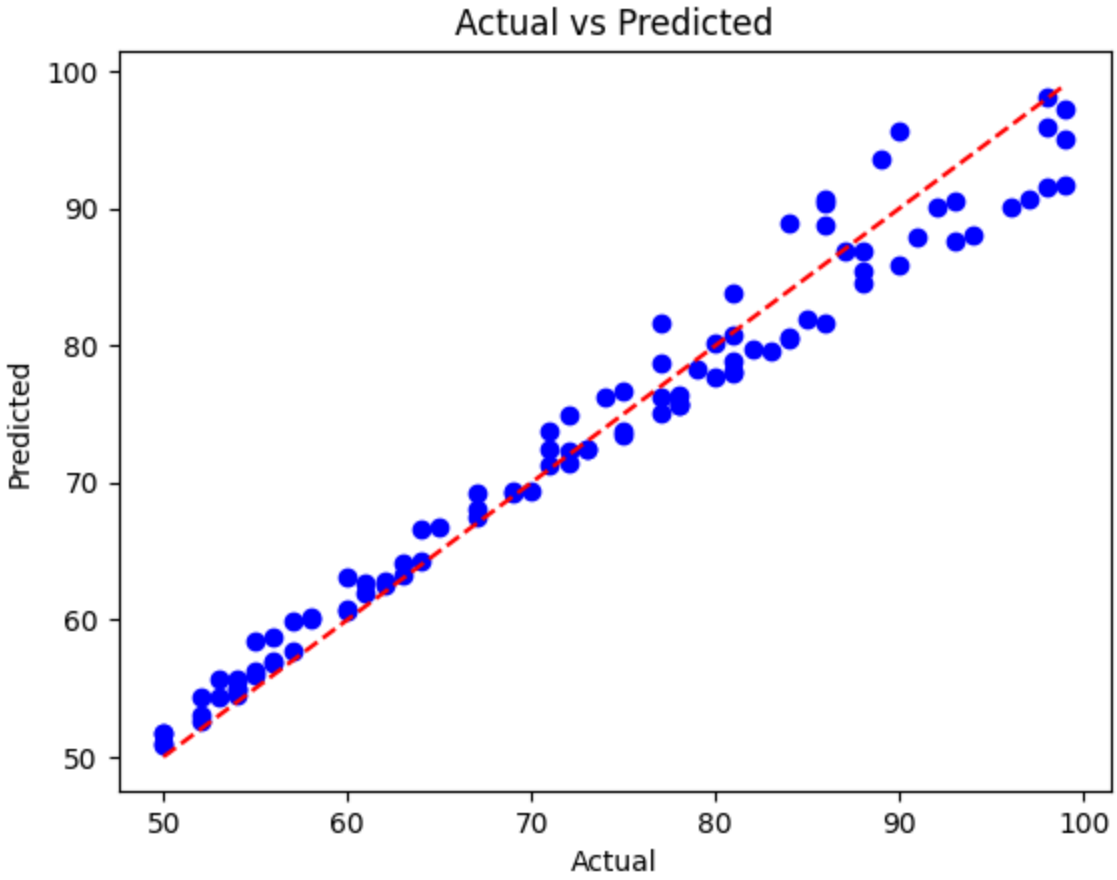
In [ ]:
```
model=Ridge()
model.fit(x_train,y_train)
prediction=model.predict(x_test)
print(prediction)
print(y_test)
```

```
[58.80006748 75.12987757 88.78295985 52.65728353 84.57419949 77.62584883
 71.27442621 68.08921491 55.95494504 72.32220856 55.6776383  81.56696548
 72.4537459  95.11125724 78.19512557 73.70558039 83.79646828 59.9374122
 90.37032074 60.24962526 75.58980431 60.7392033  55.59437082 50.9397475
 85.45015142 90.67565797 51.72675929 62.78620675 58.4558097  71.37375287
 56.22020694 80.19033471 64.11282788 64.20810023 57.04971319 62.65246145
 66.69995544 73.46685914 78.20767444 63.10436011 69.34559623 80.44617455
 73.76233324 81.96792707 54.41799456 76.23339101 88.97567185 74.89055663
 75.87275225 90.12077998 63.22805279 76.69001077 55.62757113 87.97559956
 51.80713515 79.70486343 98.15555558 72.42202455 69.23316395 97.27173089
 67.42630967 90.1629237  55.01657351 50.82727668 93.54880409 54.34166483
 75.63068424 76.454245   57.71525625 80.81942332 54.5036619  78.7893193
 79.54753471 72.42704706 78.86468265 90.49803258 85.86723307 69.38321638
 78.00686844 86.85094028 80.54626851 76.28386674 66.60368221 86.88699017
 60.11337595 91.50266087 90.66652629 95.68303773 81.5648418  54.99354972
 87.6573066  91.70052264 88.06997882 62.48530421 60.66178857 56.85551752
 62.014556   69.29397946 95.91873452 53.06569169]
[56 77 86 52 88 80 71 67 55 72 54 77 71 99 81 71 81 57 86 58 78 60 54 50
 88 86 50 62 55 72 55 80 63 64 56 61 65 75 79 60 70 84 75 85 53 74 84 72
 78 96 63 75 53 91 50 82 98 73 69 99 67 92 54 50 89 52 78 78 57 81 54 77
 83 73 81 93 90 69 81 87 84 77 64 88 58 98 97 90 86 54 93 99 94 62 60 56
 61 67 98 52]
```

In [ ]:
```
r2 = r2_score(y_test, prediction)
print("R² Score:", r2)
```

```
R² Score: 0.9654571788361487
```

In [ ]:
```
plt.scatter(y_test, prediction, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("Actual vs Predicted")
plt.show()
```


Actual vs Predicted
Predicted
Actual
100
90
80
70
60
50
50
60
70
80
90
100