1. I have used multivariable linear regression model. Linear regression performs the task to predict a dependent variable value (y) based on given independent variables in this case workout and diet. So, this regression technique finds out a linear relationship between x1,x2,x3,x4 (input) no activity,less vegetables, fuits and moderate activity respectively and y(output) obesity so here we are trying to find out the relationship between the input x1,x2,x3,x4 and output y obesity and linear model is a suitable choice for this kind of a problem.

2. https://www.kaggle.com/datasets/spittman1248/cdc-data-nutrition-physical-activity-obesity/code this is the link for the data it included features like location,state,value,gender and some more I preprocessed the data and the processed data only included the features which has effect on the output (y) in this case obesity I extracted those questions related to a person being obese in (y) and questions about people doing no activity, eating less fruits, eating less vegetables and people who do moderate activity since there is not direct relationship between the output obesity class obesity and the features x1,x2,x3,x4 I used the year feature as the link between two to find out the percentage of obesity in x1,x2,x3,x4 in a span of one year. The relationship is evaluated based on the feature “value”.

3.

```
import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

from sklearn.linear_model import Ridge

from sklearn import metrics

from sklearn.linear_model import LinearRegression

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import cross_validate

from sklearn.pipeline import make_pipeline

from sklearn.preprocessing import PolynomialFeatures

df = pd.read_csv("/content/sample_data/Nutrition__Physical_Activity__and_Obesity_-_Behavioral_Risk_Factor_Surveillance_System.csv")

df.head()

# df.describe()
```

```
#getting data of noactivity

wrangled = df[df['StratificationID1'] == 'OVERALL'][['LocationDesc','Data_Value', 'Question', "YearStart" ]]

question = wrangled[wrangled['Question'] == 'Percent of adults who engage in no leisure-time physical activity'][['LocationDesc','Data_Value', 'Question', "YearStart" ]]

activity_all = question[question['YearStart'] == 2011][['LocationDesc','Data_Value' ]]

no_activity = question[question['YearStart'] == 2011][['Data_Value' ]].values

no_activity = numpy.array(no_activity)




from scipy.linalg.decomp import numpy

#consuming vegetables

wrangled1 = df[df['StratificationID1'] == 'OVERALL'][['LocationDesc','Data_Value', 'Question', "YearStart" ]]

question = wrangled1[wrangled1['Question'] == 'Percent of adults who report consuming vegetables less than one time daily'][['LocationDesc','Data_Value', 'Question', "YearStart" ]]

veg1_all = question[question['YearStart'] == 2011][['LocationDesc','Data_Value' ]]

veg2 = question[question['YearStart'] == 2011][['Data_Value' ]].values

veg2 = numpy.array(veg2)




#consuming less fruits

wrangled1 = df[df['StratificationID1'] == 'OVERALL'][['LocationDesc','Data_Value', 'Question', "YearStart" ]]

question = wrangled1[wrangled1['Question'] == 'Percent of adults who report consuming vegetables less than one time daily'][['LocationDesc','Data_Value', 'Question', "YearStart" ]]

fruit_all = question[question['YearStart'] == 2011][['LocationDesc','Data_Value' ]]

fruits = question[question['YearStart'] == 2011][['Data_Value' ]].values
```

```
fruits = numpy.array(fruits)



#moderate_activity

wrangled = df[df['StratificationID1'] == 'OVERALL'][['LocationDesc','Data_Value', 'Question', "YearStart" ]]

question = wrangled[wrangled['Question'] == 'Percent of adults who engage in no leisure-time physical activity'][['LocationDesc','Data_Value', 'Question', "YearStart" ]]

moderate_activity = question[question['YearStart'] == 2011][['LocationDesc','Data_Value' ]]

moderate_activity = question[question['YearStart'] == 2011][['Data_Value' ]].values

moderate_activity = numpy.array(moderate_activity)



no_activity = no_activity.reshape(-1, 1)

veg2 = veg2.reshape(-1, 1)

fruits = fruits.reshape(-1, 1)

moderate_activity = moderate_activity.reshape(-1, 1)

arr = np.append(no_activity, veg2, axis=1)

arr = np.append(arr,fruits,axis=1)

arr = np.append(arr,moderate_activity,axis = 1)

arr



# the output for the input data

wrangled = df[df['StratificationID1'] == 'OVERALL'][['LocationDesc','Data_Value', 'Question', "YearStart" ]]
```

```
year = wrangled[wrangled['Question'] == 'Percent of adults aged 18 years and older who have obesity']][['LocationDesc','Data_Value', 'Question', "YearStart" ]]

y_all = year[year['YearStart'] == 2011][['LocationDesc','Data_Value' ]]

y = year[year['YearStart'] == 2011][['Data_Value' ]].values

y = y.reshape(-1, 1)
```

```
x_train, x_test, y_train, y_test = train_test_split(arr, y, train_size=.6, test_size=.4, random_state=100)

print(f'X Train Data shape{x_train.shape}')

print(f'y Train Data shape{y_train.shape}')

print(f'X Test Data shape{x_test.shape}')

print(f'y Test Data shape{y_test.shape}')
```

```
lm = LinearRegression()

lm.fit(x_train, y_train)

y_predict = lm.predict(x_test)
```

```
print(f'Train Accuracy {round(lm.score(x_train, y_train)* 100,2)}%')

print(f'Test Accuracy {round(lm.score(x_test, y_test)* 100,2)}%')
```

```
cv_results = cross_validate(lm, arr, y,
                            cv=10, scoring="neg_mean_squared_error",
                            return_train_score=True,
                            return_estimator=True)
train_error = -cv_results["train_score"]
print(f"Mean squared error of linear regression model on the train set:\n"
    f"{train_error.mean():.3f} ± {train_error.std():.3f}")



test_error = -cv_results["test_score"]
print(f"Mean squared error of linear regression model on the test set:\n"
    f"{test_error.mean():.3f} ± {test_error.std():.3f}")



ridge = Ridge()
ridge.fit(x_train, y_train)
print("Ridge Train RMSE:", np.round(np.sqrt(metrics.mean_squared_error(y_train,
ridge.predict(x_train))), 2))
print("Ridge Test RMSE:", np.round(np.sqrt(metrics.mean_squared_error(y_test, ridge.predict(x_test))),
2))
```

4. I used **regression's cost function: the mean squared error (MSE)**. Using a library mentioned in the code snippet. It can be used to test the performance of a model.

```
cv_results = cross_validate(lm, arr, y,
                            cv=10, scoring="neg_mean_squared_error",
                            return_train_score=True,
                            return_estimator=True)
train_error = -cv_results["train_score"]
print(f"Mean squared error of linear regression model on the train set:\n"
      f"{train_error.mean():.3f} ± {train_error.std():.3f}")
```

5. there was under fitting issue since there was a big difference between the training score and the test score it only performed well on training data but performed poorly on testing data. so I used Ridge or L2. It is a Regularization Technique in which the summation of squared values of the coefficients of the regression equation is added as penalty into cost function (or MSE). Ridge Regression is based on the L2 Regularization technique. The below code snippet is the working of ridge regularization technique.

```
ridge = Ridge()
ridge.fit(x_train, y_train)
print("Ridge Train RMSE:", np.round(np.sqrt(metrics.mean_squared_error(y_t
rain, ridge.predict(x_train))), 2))
print("Ridge Test RMSE:", np.round(np.sqrt(metrics.mean_squared_error(y_te
st, ridge.predict(x_test))), 2))
```