Homework 1 Part 2

This is an individual assignment.

Write your own code. You may repurpose any functions built during lecture. You may use `scikit-learn` functions.

In [1]:
```
# Import libraries and magics

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('bmh')
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.model_selection import cross_val_score
from IPython.display import Image
```

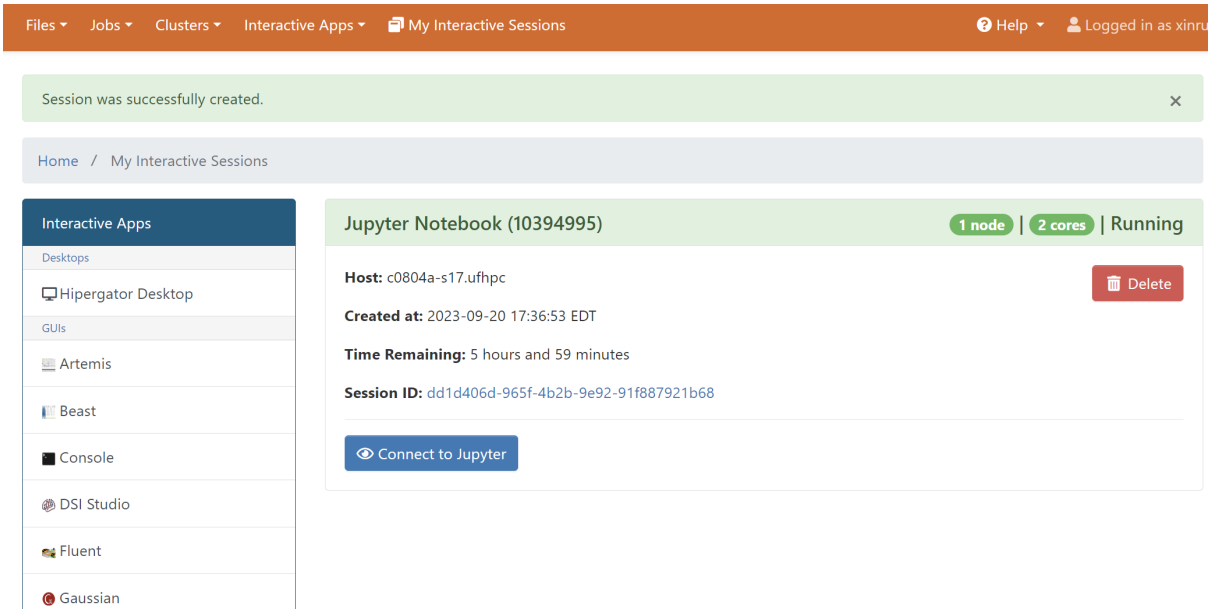
Exercise 1 (5 points)

In this question, you will practice how to use HiPerGator and Git to maintain your code.

1.(1 point) Open Open On-Demand ood.rc.ufl.edu and create an interactive jupyter session with the following specifications: 2 CPU, 2 GPU (type = 'a100') and 4 GB of RAM. **Attach a screenshot of your jupyter notebook card, your gatorlink should be visible.**

In [2]:
```
Image('1.png', width=900)
```

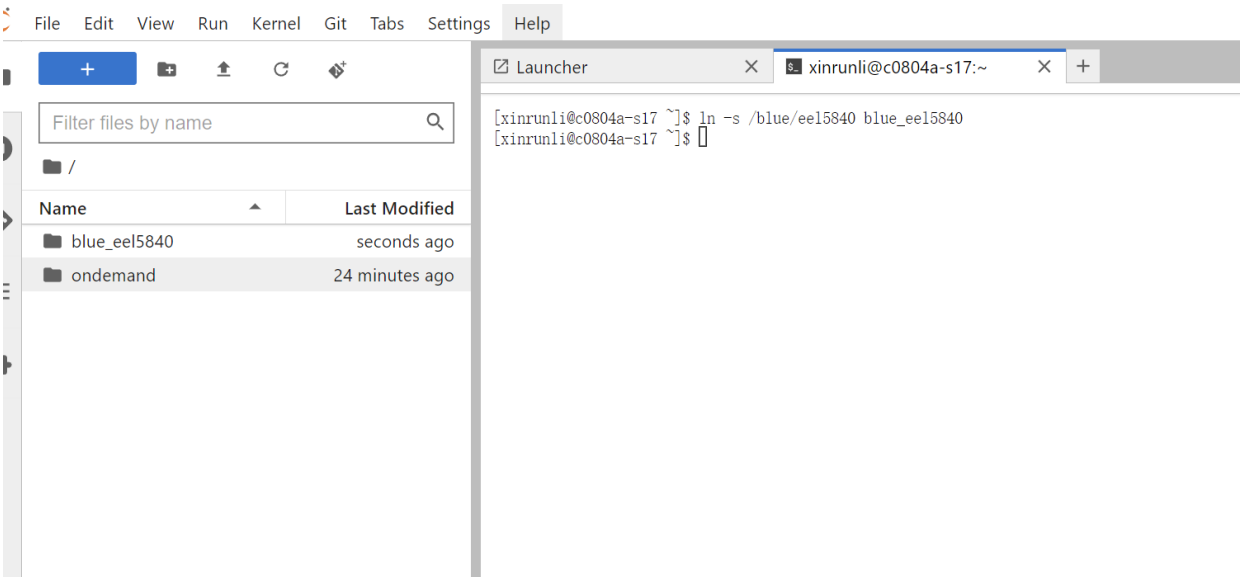
Out[2]:



2.(2 points) Create a symbolic link to map the class blue directory in your homepage. **Attach a screenshot to show that this link has been created.**

In [3]:
```
Image('2.png', width=900)
```

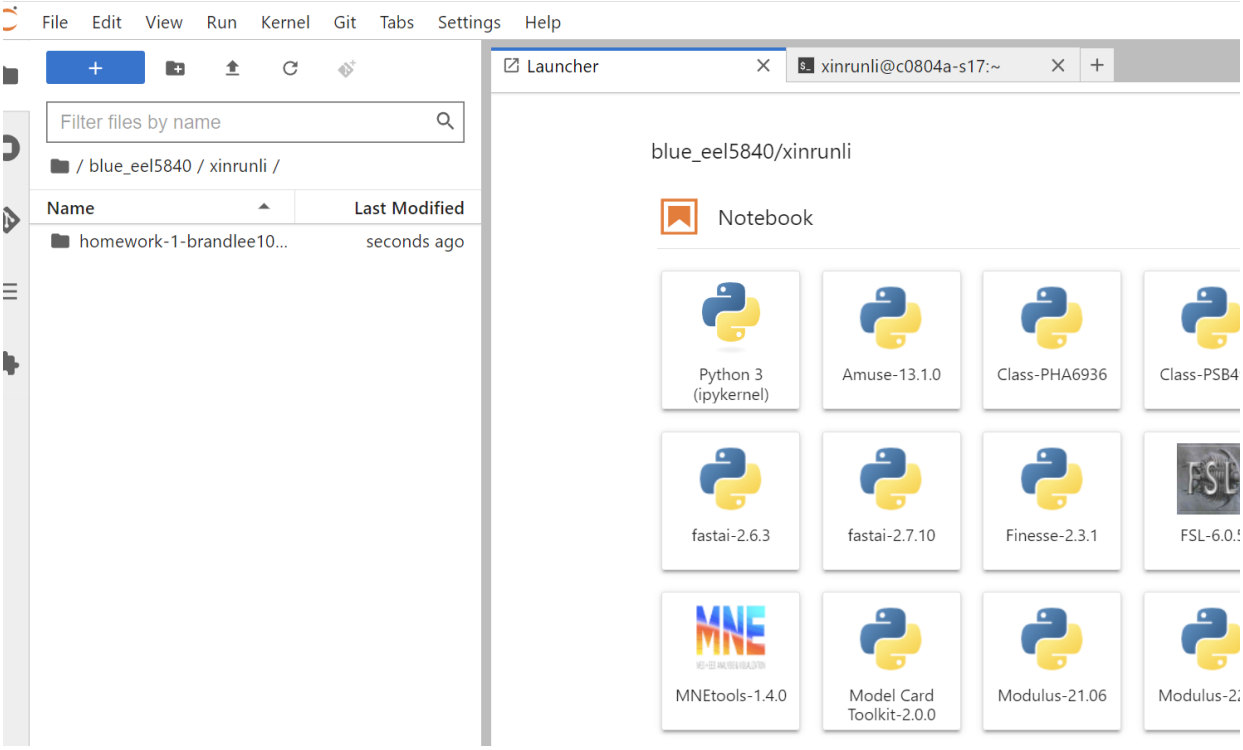
Out[3]:



3.(2 points) Navigate to your folder within the symbolic link you just created. Clone the HW0 repository inside that folder - see the picture below. If you clone outside your personal and private folder (mine is blue_aml_f23/catiaspsilva), you will lose points. **Attach a screenshot.**

In [4]:
```
Image('3.png', width=900)
```

Out[4]:



Exercise 2 (20 points)

Consider the noisy sinusoidal data we have been working with from lecture.

Build a linear regression model with Gaussian basis functions as feature representations of the data. Consider the Gaussian basis functions:

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2)}{2\sigma^2}\right\}$$

where $\mu = \{0.1, 0.3, 0.6, 0.9\}$ for $j = 1, 2, 3, 4$, respectively, and a fixed standard deviation $\sigma = 0.1$.

1. (7 points) **Train this model using the training set generated below.**
2. (7 points) **Make predictions using the test set.**
3. (6 points) **Provide a paragraph discussion about how you would determine how many Gaussian basis functions you would need and how would you determine the mean values μ_j and the bandwidth parameter σ.**

In [5]:

```
def NoisySinusoidalData(N, a, b, sigma):
    '''Generates N data points in the range [a,b) sampled from a sin(2*pi*x)
    with additive zero-mean Gaussian random noise with standard deviation sigma'''

    # N input samples, evenly spaced numbers between [a,b) incrementing by 1/N
    x = np.linspace(a,b,N)

    # draw N sampled from a univariate Gaussian distribution with mean 0, sigma standar
```

```
    noise = np.random.normal(0,sigma,N)

    # desired values, noisy sinusoidal
    t = np.sin(2*np.pi*x) + noise

    return x, t
```

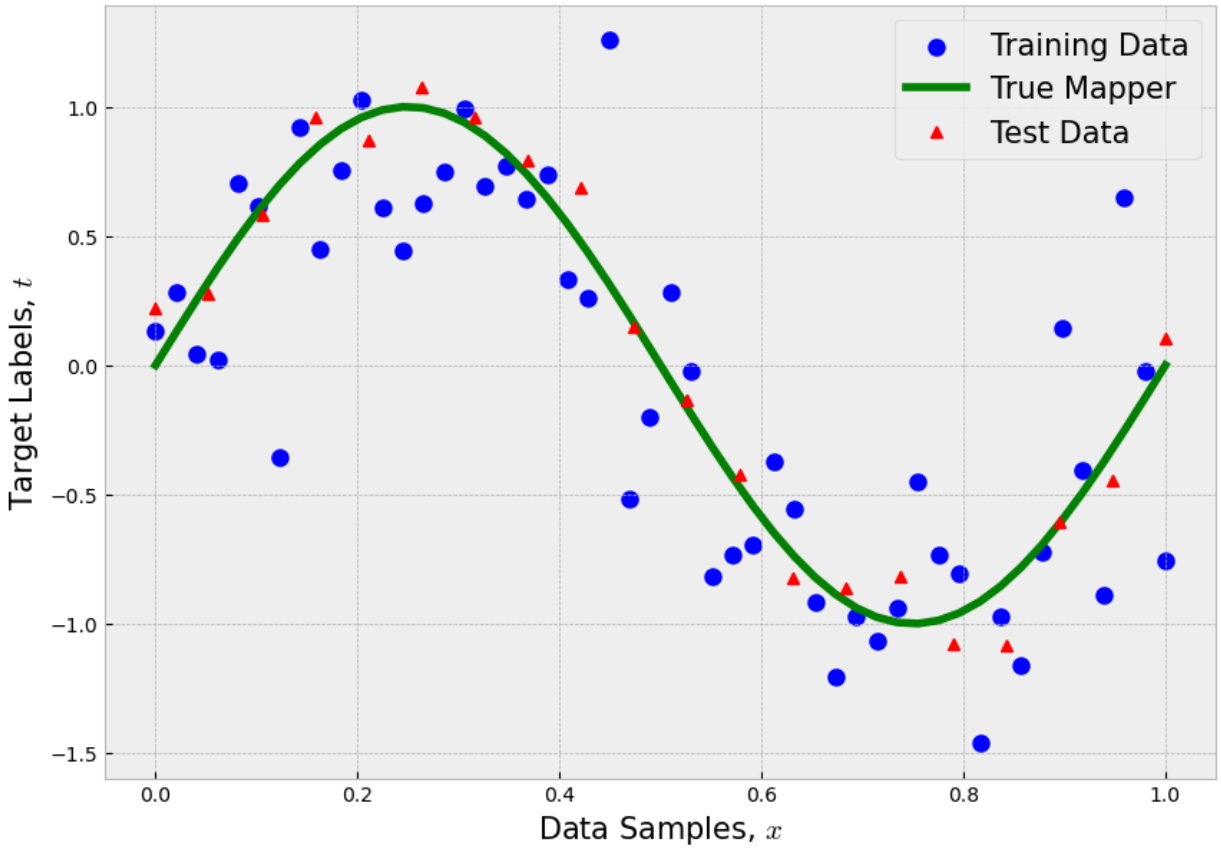
In [6]:
```
# Generate input samples and desired values
N_train = 50 # number of data samples for training
N_test = 20 # number of data samples for test

a, b = [0,1] # data samples interval

sigma_train = 0.4 # standard deviation of the zero-mean Gaussian noise -- training data
sigma_test = 0.1 # standard deviation of the zero-mean Gaussian noise -- test data

x_train, t_train = NoisySinusoidalData(N_train, a, b, sigma_train) # Training Data - No
x_true, t_true = NoisySinusoidalData(N_train, a, b, 0) # True Sinusoidal - in practice,
x_test, t_test = NoisySinusoidalData(N_test, a, b, sigma_test) # Test Data - Noisy sinu

# Plotting
plt.figure(figsize=(10,7))
plt.scatter(x_train, t_train, c='b', linewidths=3, label = 'Training Data')
plt.plot(x_true, t_true, 'g', linewidth=4, label = 'True Mapper')
plt.plot(x_test, t_test, 'r^', label = 'Test Data')
plt.legend(fontsize=15)
plt.xlabel('Data Samples, $x$',size=15)
plt.ylabel('Target Labels, $t$',size=15);
```



1.Train this model using the training set generated below

In [7]:
```
def gaussian_basis(x, mu, sigma):
    return np.exp(-((x - mu) ** 2) / (2 * sigma ** 2))

mus = [0.1, 0.3, 0.6, 0.9]
sigma = 0.1

design_matrix = np.zeros((N_train, len(mus)))

for j, mu in enumerate(mus):
    design_matrix[:, j] = gaussian_basis(x_train, mu, sigma)


model = LinearRegression()
model.fit(design_matrix, t_train)
```

Out[7]:

▾ LinearRegression

LinearRegression()

2.Make predictions using the test set.

In [8]:
```
test_design_matrix = np.zeros((N_test, len(mus)))

for j, mu in enumerate(mus):
    test_design_matrix[:, j] = gaussian_basis(x_test, mu, sigma)

predictions = model.predict(test_design_matrix)
print(predictions)
```

```
[ 0.10903694  0.23083497  0.36151104  0.51978128  0.71977805  0.88573019
  0.87652736  0.63392616  0.24262291 -0.1703968  -0.51390155 -0.71291441
 -0.72350208 -0.60354048 -0.49793363 -0.5119412  -0.61429953 -0.67688907
 -0.60888913 -0.44007895]
```

3.Provide a paragraph discussion about how you would determine how many Gaussian basis functions you would need and how would you determine the mean values μj and the bandwidth parameter σ

Determing the number of basis functions: we can use cross-validation to determine how many basis functions should be applied by selecting model with best performance.
Determing mean values μj and the bandwidth parameter σ:
1.we can use grid search with cross-validation to find optimal μj and σ.
2.Apply regularization to control μj and σ and prevent overfitting.

In [ ]:
```
```

Exercise 3 (20 points)

Consider the diabetes data:

In [9]:
```
from sklearn.datasets import load_diabetes

diabetes = load_diabetes(return_X_y=False)
```

This dataset is already described in the *feature space*. Each input sample x_i is described as 10-dimensional feature vctor $\phi(x_i)$. The features correspond to: age, sex, bmi, bp, s1, s2, s3, s4, s5 and s6 measurements (read the description above for more details). The target variable corresponds a measure of diabetes disease progression one year after baseline.

Let's load the data as a `pandas` dataframe:

In [10]:
```
df_diabetes = pd.DataFrame(data=np.hstack((diabetes.target[:,np.newaxis],diabetes.data)
                  columns=['Target']+diabetes.feature_names)

df_diabetes
```

Out[10]:

	Target	age	sex	bmi	bp	s1	s2	s3	s4	
0	151.0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.
1	75.0	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.
2	141.0	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.
3	206.0	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.
4	135.0	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.
...	...	...	...	...	...	...	...	...	...	...	
437	178.0	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.
438	104.0	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.
439	132.0	0.041708	0.050680	-0.015906	0.017293	-0.037344	-0.013840	-0.024993	-0.011080	-0.
440	220.0	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.
441	57.0	-0.045472	-0.044642	-0.073030	-0.081413	0.083740	0.027809	0.173816	-0.039493	-0.

442 rows × 11 columns

In [11]:
```
df_diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 442 entries, 0 to 441
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Target  442 non-null    float64
 1   age     442 non-null    float64
 2   sex     442 non-null    float64
 3   bmi     442 non-null    float64
 4   bp      442 non-null    float64
 5   s1      442 non-null    float64
 6   s2      442 non-null    float64
 7   s3      442 non-null    float64
 8   s4      442 non-null    float64
 9   s5      442 non-null    float64
 10  s6      442 non-null    float64
dtypes: float64(11)
memory usage: 38.1 KB
```

The goal is to fit a linear regression model on the provided features, i.e., the model is of the form:

$$y(x) = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + w_3\phi_3(x) + \cdots + w_{10}\phi_{10}(x)$$

where w_0 is the bias (or intercept) coefficient and all other $w_i, i = 1, \ldots, 10$ correspond to the coefficient associated with feature ϕ_i (age, sex, bmi, bp, etc.).

Answer the following questions:

1. (2 points) **Randomly partition the data into training (70%) and test sets (30%) with a fixed random seed generator.**
2. (6 points) **Use a 5-fold cross-validation strategy to determine the hyperparameter values to fit a linear regression model with ridge regularization for this dataset. Show and document your work.**
3. (4 points) **Evaluate performance in the test set.**
4. (4 points) **Determine the final value for the intercept and coefficients of the linear regression model. Plot all 11 values as a `stem` plot.**
5. (4 points) **Based on this plot, which input variable (also referred to the independent variable) has the most contribution for predicting the target variable (also referred to the dependent variable)?**

1.Randomly partition the data into training (70%) and test sets (30%) with a fixed random seed generator.

In [12]:
```
X = df_diabetes.iloc[:, 1:]
y = df_diabetes['Target']

random_seed = 42
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=r
```

2.Use a 5-fold cross-validation strategy to determine the hyperparameter values to fit a linear regression model with ridge regularization for this dataset. Show and document your work.

In [13]:
```
alphas = np.logspace(-6, 6, 13)

cv_scores = []

for alpha in alphas:

    model = Ridge(alpha=alpha)

    scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_error')

    cv_scores.append(-np.mean(scores))

optimal_alpha = alphas[np.argmin(cv_scores)]
min_mse = min(cv_scores)

print(f"Optimal Alpha: {optimal_alpha}")
print(f"Minimum Mean Squared Error (MSE): {min_mse}")
```

```
Optimal Alpha: 0.0001
Minimum Mean Squared Error (MSE): 2993.046068249655
```

3.Evaluate performance in the test set.

In [14]:
```
ridge_model = Ridge(alpha=optimal_alpha)
ridge_model.fit(X, y)

y_pred = ridge_model.predict(X_test)


mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse}")
```

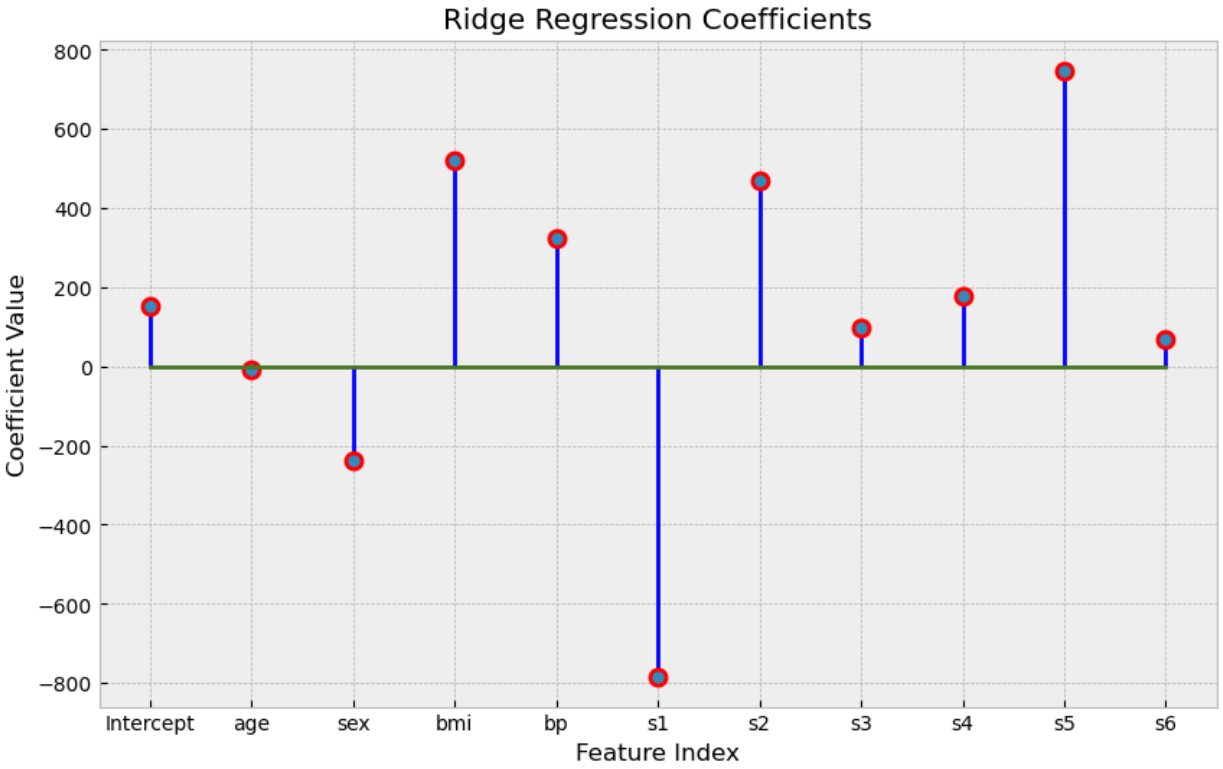
```
Mean Squared Error (MSE): 2634.438925956755
```

4.Determine the final value for the intercept and coefficients of the linear regression model. Plot all 11 values as a stem plot.

In [15]:
```
intercept = ridge_model.intercept_
coefficients = ridge_model.coef_

feature_names = ["age", "sex", "bmi", "bp", "s1", "s2", "s3", "s4", "s5", "s6"]
plt.figure(figsize=(10, 6))
markerline, stemline, baseline = plt.stem(range(1, 12), [intercept] + list(coefficients
plt.setp(markerline, marker='o', markersize=8, markeredgecolor="red", markeredgewidth=2
plt.setp(stemline, color="blue")
plt.setp(baseline, visible=True)
plt.xlabel("Feature Index")
plt.ylabel("Coefficient Value")
plt.title("Ridge Regression Coefficients")
plt.xticks(range(1, 12), ["Intercept"] + feature_names)
plt.grid(True)
plt.show()
```

```
print(f"Intercept (Bias): {intercept}")
print("Coefficients:")
for i, coef in enumerate(coefficients):
    print(f"Feature {feature_names[i]}: {coef}")
```




```
Intercept (Bias): 152.13348416289597
Coefficients:
Feature age: -9.959966799830068
Feature sex: -239.73847277123187
Feature bmi: 519.9079015835796
Feature bp: 324.3246983983759
Feature s1: -783.360954289244
Feature s2: 469.74463276365964
Feature s3: 97.14958562971226
Feature s4: 176.0030790377875
Feature s5: 747.9310579824976
Feature s6: 67.67944395500378
```

5.Based on this plot, which input variable (also referred to the independent variable) has the most contribution for predicting the target variable (also referred to the dependent variable)?

In this case, the feature with the highest absolute coefficient value in the plot is "s1", which has the most contribution for predicting the target variable.

On-Time (5 points)

Submit your assignment before the deadline.

Submit Your Solution

Confirm that you've successfully completed the assignment.

Along with the Notebook, include a PDF of the notebook with your solutions.

`add` and `commit` the final version of your work, and `push` your code to your GitHub repository.

Submit the URL of your GitHub Repository as your assignment submission on Canvas.
