Short Assignment 1

This is an individual assignment.

Objectives

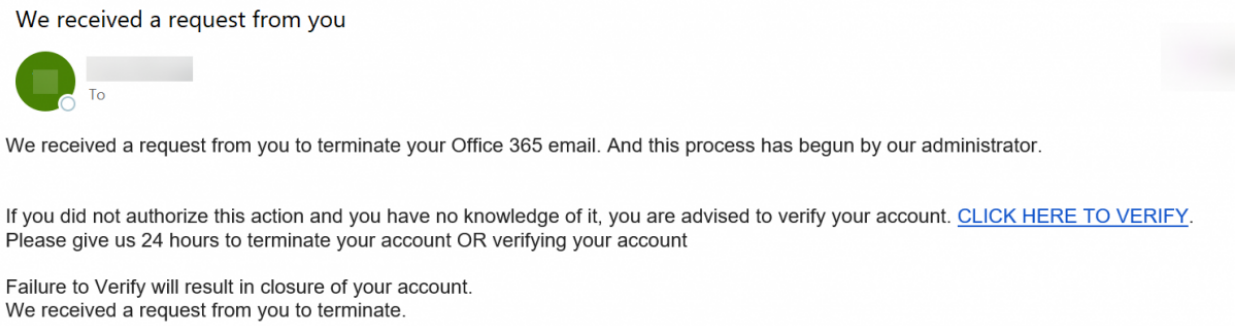
- Define basic terminology in supervised learning algorithms.
- Develop a supervised learning algorithm for regression tasks.
- Implement linear regression model with Python code.
- Utilize trained model for making predictions and measure performance.

Question 1 (1 point)

Assume we are given the task of building a system to distinguish an email as spam or ham (not spam).

In [1]:
```
from IPython.display import Image
Image('figures/spam_email.png', width=900)
```

Out[1]:



Use the fundamental components of a supervised learning system to illustrate how you would design a system to solve this task. Explain the role of each component in the system in accomplishing the task of classifying an email as spam or ham.

Trainning Stage

1. Get a dataset with labels(in this example, the emails are distinguished by spam/ham)
2. Extract features(convert email text into numerical features that machine learning algorithms can work with) and define feature space, mapper and objective function
3. Train the model(the model can learn if a email is spam or not with given labels) ## Testing Stage

4. Test data without labels
5. Extract (the same) features(we extract features from the testing data in a same way so that they are compatible with the model we trainned)
6. Use trainned model to make predictions(the result can be used to access the availability of the clarification model)

In [ ]:

In [ ]:

Question 2 (1.5 points)

Consider the training set with N data points $\{x_i\}_{i=1}^N$, where $x_i \in \mathbb{R}$, and its corresponding target labels $\{t_i\}_{i=1}^N$, where $t_i \in \mathbb{R}$. Consider the feature representation

$$\phi(x) = [1, \phi_1(x), \phi_2(x), \ldots, \phi_M(x)]^T$$

where $\phi_i(x)$ is a Gaussian basis function defined as:

$$\phi_i(x) = \exp(-\gamma_i(x - \mu_i)^2), \quad i = 1, \ldots, M$$

with μ_i as the center of component i and γ_i the precision (inverse of variance) of component i.

Answer the following questions:

1. (0.5 points) **Write down the feature matrix $\mathbf{X}$ for $M = 2$. Keep your notation neat.**

$$[1, exp(-\gamma_1(x - \mu_1)^2), exp(-\gamma_2(x - \mu_2)]^T$$

In [ ]:

1. (0.5 points) **What are the hyperparameters in this problem?**

μi,γi and M

In [ ]:

In [ ]:

1. (0.5 points) **Suppose you want to minimize the *squared error* with a *Lasso regularizer*. Write down the objective function.**

$$J(w) = \frac{1}{2}\sum_{n=1}^{N}[\phi(x_n, w) - t_n] + \sum_{j=0}^{M}|w_j|$$

In [ ]:

In [ ]:

Question 3 (1 point)

Suppose that you are working with a two-dimensional features space, where x_1 and x_2 are the two features. Upon receiving a sample $\mathbf{x} = [x_1, x_2]^T$, the goal is to predict a continuous value t. Assume that we have examples of training pairs $\{(x_1, x_2)_i, t_i\}_{i=1}^{N}$. Suppose we want to make a quadratic mapper function of the form,

$$f(x_1, x_2) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_4(x_1)^2 + w_5(x_2)^2$$

Explain how you can find an analytical solution for $w_i \;\; i = 0, 1, \ldots, 5$.

Firstly, we write down the objection function

$$J(w) = \frac{1}{2}\sum_{i=1}^{N}[f(x_{1i}, x_{2i}) - t_i]^2$$

Since it is a quadratic function of the coefficient w, its derivatives with respect to the coefficients will be linear in the elements of w. Therefore,the minimization of the MSE has a unique solution, namely the analytical solution we are looking for

$$w_i(i = 0, 1, \ldots, 5.)$$

In [ ]:

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

Question 4 (3 points)

In this problem, you will be working with the beer dataset with information about the foam height (in cm) from 3 brands of beer over 15 measurement times (in seconds) after the time of pour.

In [2]:
```
import pandas as pd

beer_data = pd.read_csv('beer_foam.csv')

beer_data
```

Out[2]:

	Time	Erdinger	Augustinerbrau	Budweiser
0	0	17.0	14.0	14.0
1	15	16.1	11.8	12.1
2	30	14.9	10.5	10.9
3	45	14.0	9.3	10.0
4	60	13.2	8.5	9.3
5	75	12.5	7.7	8.6
6	90	11.9	7.1	8.0
7	105	11.2	6.5	7.5
8	120	10.7	6.0	7.0
9	150	9.7	5.3	6.2
10	180	8.9	4.4	5.5
11	210	8.3	3.5	4.5
12	240	7.5	2.9	3.5
13	300	6.3	1.3	2.0
14	360	5.2	0.7	0.9

Consider the first 12 samples as the training set, and the last 3 samples as the test set.

In [49]:
```
# Training and test sets
x_train = beer_data['Time'].to_numpy()[:12]
x_test  = beer_data['Time'].to_numpy()[12:]

# Training and test labels
t_train = beer_data.drop('Time', axis=1).iloc[:12]
t_test  = beer_data.drop('Time', axis=1).iloc[12:]
# Note that t_train and t_test contain 3 target vectors.
```

Use the Python code implementation we built in class to help you train a mapper function of the form:

$$y(x) = \exp\left(\sum_{j=0}^{M} w_j x^j\right) = \exp(\mathbf{X}\mathbf{w})$$

Answer the following questions:

1. (1 point) **For each brand, train a mapper function with $M = 2$ using the training data. Plot the model prediction.**

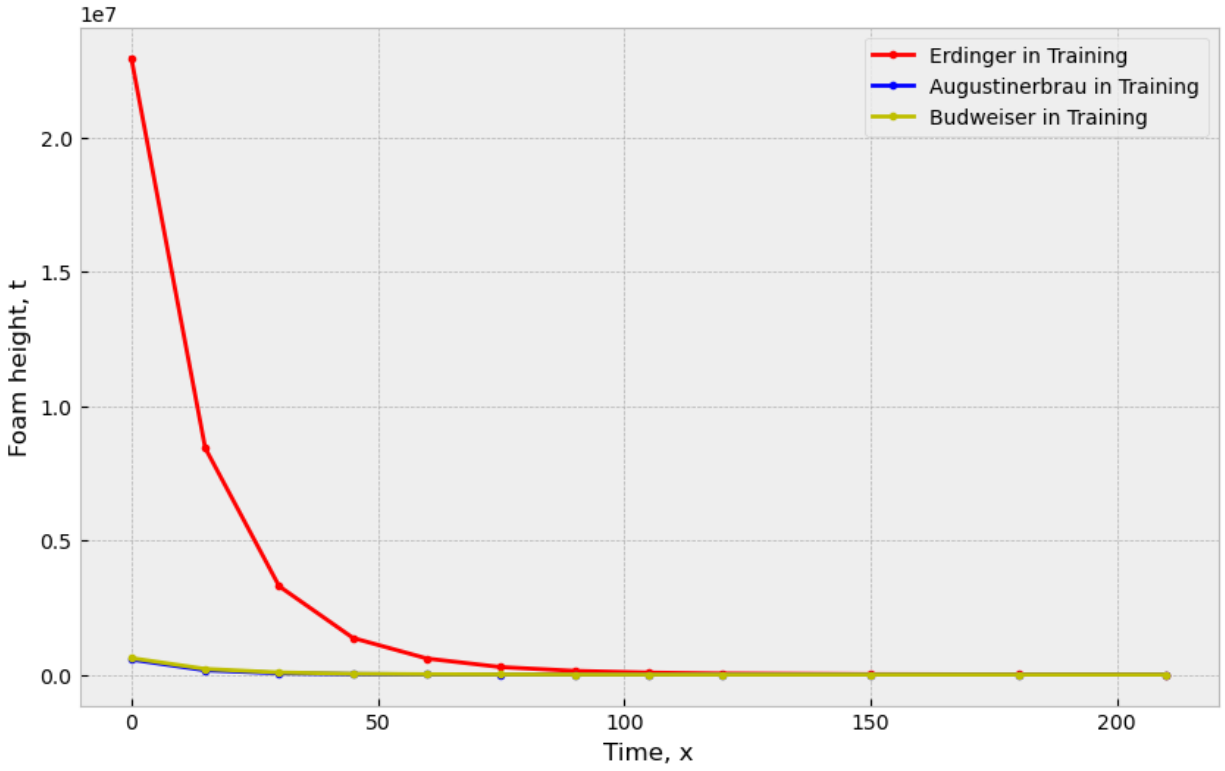
2. (1 point) **Use each trained model to make predictions for the test data.**

3. (1 point) **For each brand, predict foam height at $t = 450$ seconds.**

In [5]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('bmh')
```

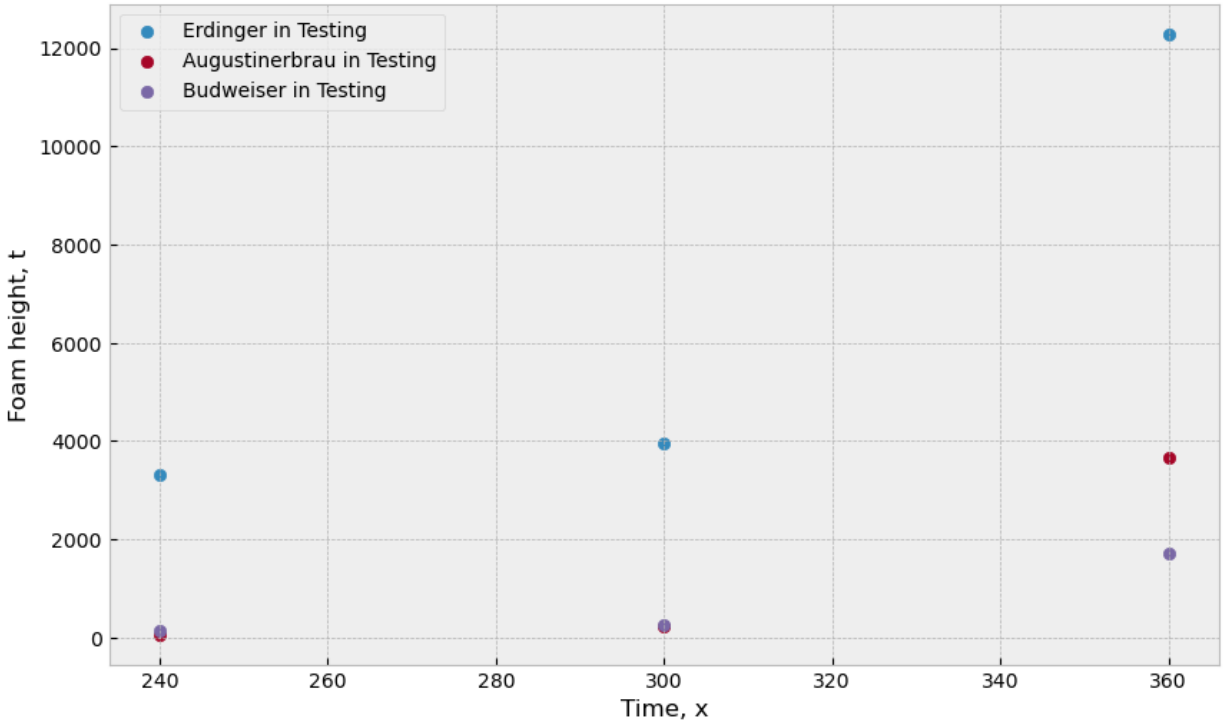
In [55]:
```
def model(x,t,M):
    X = np.array([x**j for j in range(M+1)]).T
    w = np.linalg.inv(X.T@X)@X.T@t
    y = np.exp(X@w)
    return w,y
```

In [56]:
```
M = 2
w1,y_train1 = model(x_train, t_train['Erdinger'], M)
w2,y_train2 = model(x_train, t_train['Augustinerbrau'], M)
w3,y_train3 = model(x_train, t_train['Budweiser'], M)

plt.figure(figsize=(10,6))
plt.plot(x_train, y_train1, '.-r', label = 'Erdinger in Training')
plt.plot(x_train, y_train2, '.-b', label = 'Augustinerbrau in Training')
plt.plot(x_train, y_train3, '.-y', label = 'Budweiser in Training')
plt.legend()
plt.xlabel('Time, x')
plt.ylabel('Foam height, t');
```



In [51]:
```
def model_test(x_test,w):
    X = np.array([x_test**j for j in range(len(w))]).T
    y = np.exp(X@w)
    return y
```

In [61]:
```
y_test1 = model_test(x_test, w1)
y_test2 = model_test(x_test, w2)
y_test3 = model_test(x_test, w3)

plt.figure(figsize=(10,6))
plt.scatter(x_test, y_test1, label = 'Erdinger in Testing')
plt.scatter(x_test, y_test2, label = 'Augustinerbrau in Testing')
plt.scatter(x_test, y_test3, label = 'Budweiser in Testing')
plt.legend()
plt.xlabel('Time, x')
plt.ylabel('Foam height, t');
```



In [67]:
```
x_predict = 450

# Use the w values from your trained models to make predictions
y_predict1 = np.exp(np.array([x_predict**j for j in range(M+1)]) @ w1)
y_predict2 = np.exp(np.array([x_predict**j for j in range(M+1)]) @ w2)
y_predict3 = np.exp(np.array([x_predict**j for j in range(M+1)]) @ w3)

# Print or use the predicted foam heights
print("Predicted foam height for Erdinger at t=450:", y_predict1)
print("Predicted foam height for Augustinerbrau at t=450:", y_predict2)
print("Predicted foam height for Budweiser at t=450:", y_predict3)
```

```
Predicted foam height for Erdinger at t=450: 405250.3981504856
Predicted foam height for Augustinerbrau at t=450: 3654146.8134749536
Predicted foam height for Budweiser at t=450: 240027.05600654095
```

In [ ]:

Question 5 (2.5 points)

Consider the [computer hardware dataset**]**
(https://archive.ics.uci.edu/ml/datasets/Computer+Hardware). The goal is to predict the estimated relative performance (ERP) of a CPU core as a function of 9 features (or independent variables):

- **Vendor name:** 30 (adviser, amdahl,apollo, basf, bti, burroughs, c.r.d, cambex, cdc, dec, dg, formation, four-phase, gould, honeywell, hp, ibm, ipl, magnuson, microdata, nas, ncr, nixdorf, perkin-elmer, prime, siemens, sperry, sratus, wang)
- **Model Name:** many unique symbols
- **MYCT:** machine cycle time in nanoseconds (integer)
- **MMIN:** minimum main memory in kilobytes (integer)
- **MMAX:** maximum main memory in kilobytes (integer)
- **CACH:** cache memory in kilobytes (integer)
- **CHMIN:** minimum channels in units (integer)
- **CHMAX:** maximum channels in units (integer)
- **PRP:** published relative performance (integer)

And the target is:

- **ERP:** estimated relative performance from the original article (integer).

In [1]:
```
import pandas as pd

hardware=pd.read_csv('machine.data',
                     names=['Vendor','Model Name','MYCT','MMIN','MMAX','CACH','CHMIN','CHMA
hardware
```

Out[1]:

	Vendor	Model Name	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	PRP	ERP
0	adviser	32/60	125	256	6000	256	16	128	198	199
1	amdahl	470v/7	29	8000	32000	32	8	32	269	253
2	amdahl	470v/7a	29	8000	32000	32	8	32	220	253
3	amdahl	470v/7b	29	8000	32000	32	8	32	172	253
4	amdahl	470v/7c	29	8000	16000	32	8	16	132	132
...	...	...	...	...	...	...	...	...	...	...
204	sperry	80/8	124	1000	8000	0	1	8	42	37
205	sperry	90/80-model-3	98	1000	8000	32	2	8	46	50
206	sratus	32	125	2000	8000	0	2	14	52	41
207	wang	vs-100	480	512	8000	32	0	0	67	47
208	wang	vs-90	480	1000	4000	0	0	0	45	25

209 rows × 10 columns

In [68]:
```
# Feature matrix
data = hardware.drop('ERP', axis=1)
data
```

Out[68]:

	Vendor	Model Name	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	PRP
0	adviser	32/60	125	256	6000	256	16	128	198
1	amdahl	470v/7	29	8000	32000	32	8	32	269
2	amdahl	470v/7a	29	8000	32000	32	8	32	220
3	amdahl	470v/7b	29	8000	32000	32	8	32	172
4	amdahl	470v/7c	29	8000	16000	32	8	16	132
...	...	...	...	...	...	...	...	...	...
204	sperry	80/8	124	1000	8000	0	1	8	42
205	sperry	90/80-model-3	98	1000	8000	32	2	8	46
206	sratus	32	125	2000	8000	0	2	14	52
207	wang	vs-100	480	512	8000	32	0	0	67
208	wang	vs-90	480	1000	4000	0	0	0	45

209 rows × 9 columns

In [69]:
```
# Target labels
target = hardware['ERP']
target
```

Out[69]:
```
0      199
1      253
2      253
3      253
4      132
      ...
204     37
205     50
206     41
207     47
208     25
Name: ERP, Length: 209, dtype: int64
```

Consider only the numerical features: MYCT, MMIN, MMAX, CACH, CHMIN, CHMAX, and PRP.

Answer the following questions:

1. (0.5 points) **Partition the data randomly using an 80/20 split. See sklearn.model_selection.train_test_split.**
2. (1 point) **Train a multivariate regression model using the training data.**
3. (1 point) **Make predictions in the test set and report performance.**

In [80]:
```
from sklearn.model_selection import train_test_split

# X_train: Training data (features)
# X_test: Testing data (features)
# y_train: Training target (ERP)
# y_test: Testing target (ERP)

X_train, X_test, y_train, y_test = train_test_split(data[['MYCT', 'MMIN', 'MMAX', 'CACH
```

In [81]:
```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

Out[81]:
```
▾ LinearRegression
LinearRegression()
```

In [82]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"MAE: {mae:.2f}")
print(f"MSE: {mse:.2f}")
print(f"R2 Score: {r2:.2f}")
```

```
MAE: 25.52
MSE: 3007.89
R2 Score: 0.94
```

In [ ]:

On-Time (1 point)

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.
