DS5220: Supervised Machine Learning and Learning Theory Fall 2022

Problem Set 1

Instructor: Hongyang Ryan Zhang *Due:* ***September 30, 2022, 11:59pm***

Instructions:

- You are expected to write up the solution on your own. Discussions and collaborations are encouraged; remember to mention any fellow students you discussed with when you turn in the solution.
- There are up to three late days for all the problem sets and project submissions. Use them wisely. After that, the grade depreciates by 20% for every extra day. Late submissions are considered case by case. Please reach out to the instructor if you cannot meet the deadline.
- Submit your written solutions to Gradescope and upload your code to Canvas. You are recommended to write up the solution in LaTeX.

Collaboration: Adwait Patil, Abhinav Nippani

Name: Ayush Patel
NUID: 002765119

Problem 1 (20 points)

(a) (1 point) Calculate $Var(X)$ when X represents the outcome when a fair coin flip (i.e., $E[X] = 1/2$).

Solution:
We know that $Var(X) = E[X^2] - (E[X])^2$,
Let 'n' be the number of chances of getting Heads and 'p' be the probability of a fair coin flip i.e. (1/2),
We know that E[X]= n.p and E[X] = 1/2 ....(Given)
Therefore, $Var(X) = n^2.p - (n.p)^2$
$= ((0)^2.(1/2) + (1)^2.(1/2)) - (1/2)^2$
$= (1/2) - (1/4)$
$Var(X) = 1/4$

(b) (1 point) Find the expected value of the sum obtained when n fair coin flips are rolled independently.

Solution:

Let 'n' be the number of chances of getting Heads and 'p' be the probability of a fair coin flip i.e. (1/2),

We know that E[X]= n.p,

Therefore, Expected value = E[X] = n/2,

(c) (2 point) For three events A, B, and C, we know that: A and C are independent, B and C are independent, A and B are disjoint, $P(A \cup C) = 2/3$, $P(B \cup C) = 3/4$, $P(A \cup B \cup C) = 11/12$. Find $P(A)$, $P(B)$ and $P(C)$.

Solution:

Given: $P(A \cup C) = 2/3$, $P(B \cup C) = 3/4$ and $P(A \cup B \cup C) = 11/12$

Formulae: $P(A \cup B \cup C)$ = P(A) + P(B) + P(C) - $P(A \cap B)$ - $P(B \cap C)$ - $P(A \cap C)$ + $P(A \cap B \cap C)$ ....(1)

$P(A \cup C)$ = P(A) + P(C) - $P(A \cap C)$ ....(2)

$P(B \cup C)$ = P(B) + P(C) - $P(B \cap C)$ ....(3)

Substituting in equation (2),

2/3 = P(A) + P(C) - $P(A \cap C)$

Therefore, $P(A \cap C)$ = P(A) + P(C) - 2/3 ....(4)

Substituting in equation (3),

3/4 = P(B) + P(C) - $P(B \cap C)$

Therefore, $P(B \cap C)$ = P(B) + P(C) - 3/4 ....(5)

Now substituting (4) and (5) in (1),

Here, we also know that A and B are disjoint so $P(A \cap B)$ is NULL.

11/12 = P(A) + P(B) + P(C) - 0 - [P(A) + P(C) - 2/3] - [P(B) + P(C) - 3/4]

P(C) = 11/12 - 2/3 - 3/4

P(C) = 1/2

Using (1), we know that $P(A \cap C)$ = P(A).P(C) because the events are independent,

$P(A \cap C)$ = P(A) + P(C) - P(A).P(C)

2/3 = P(A) + 1/2 - P(A).(1/2)

2/3 - 1/2 = P(A) [1-(1/2)]

P(A) = 2/6

P(A) = 1/3

Using (1), we know that $P(B \cap C)$ = P(B).P(C) because the events are independent,

$P(B \cap C)$ = P(B) + P(C) - P(B).P(C)

3/4 = P(B) + 1/2 - P(B).(1/2)

3/4 - 1/2 = P(B) [1-(1/2)]

P(B) = 2/4

P(B) = 1/2

(d) (2 points) Consider a test to detect a disease (e.g., COVID-19), assuming that 0.6% of the population has it. The test is 97% effective in detecting an infected person. However, the test gives a false-positive result in 1% of cases (meaning that it shows a positive result if the person is not infected). What is the probability that a person gets a negative test result?

Solution:

Lets assume the total population is 1,000,000.

Assuming that the 0.6% is infected so that the total infected people are 6000.

Since the test is 97% effective in detecting an infected person, there are 5820.

Therefore, the people that are negative but Infected would be 6000 - 5820 = 180.

Not Infected people are: Total population - Total Infected = 1000000 - 6000 = 994000.

The test also gives false-positive results of 1% of 994000 which is 9940.

Therefore, the false-negative results are 994000 - 9940 = 984060.

Total positive people are 5820 + 9940 = 15760.

Total negative people are 180 + 984060 = 984240.

People	Infected	Not Infected	Total
Positive	5820	9940	15760
Negative	180	984060	984240
Total	6000	994000	1000000

P(Negative test result) = Total negative / Total population

P(Negative test result) = 984240 / 1000000

P(Negative test result) = 0.98424

(e) (2 points) If a person tests positive for the disease, what is the probability that they actually have COVID?

Solution:

From the above table we can conclude that,

P(Tests Positive and actually have COVID) = Positive infected / Total Positive

P(Tests Positive and actually have COVID) = 5820 / 15760

P(Tests Positive and actually have COVID) = 0.36928

(f) (2 points) If a person tests negative for the disease, what is the probability that they are infected with COVID?

Solution:

From the above table we can conclude that,

P(Tests Negative and having COVID) = Negative infected / Total Negative

P(Tests Negative and having COVID) = 180 / 984240
P(Tests Negative and having COVID) = 0.00018

Along with the tests, data regarding the number of symptoms shown by the patients was also recorded and is given below. The data was collected from 2 different sources.

No. of Symptoms	Patients
1	20
2	20
3	20
4	20

No. of Symptoms	Patients
1	70
2	15
3	10
4	5

(g) (2 points) Suppose you pick one patient from each of the above 2 sources independently. What would be the expected number of symptoms detected in each of them?

Solution:

Let 'n' be the number of chances of selecting a symptom and 'p' be the probability of a selecting one patient from a table i.e. (1/4),

We know that E[X]= $\sum_{i=1}^{4} n.P(X = i)$,

For the first table,

= (1).(20/80) + (2).(20/80) + (3).(20/80) + (4).(20/80)

= 2.5

Now for the second table

= (1).(70/100) + (2).(15/100) + (3).(10/100) + (4).(5/100)

= 1.5

(h) (2 points) Prove that $Var(X) = \mathbb{E}[X]^2 - (\mathbb{E}[X])^2$. Explain the interpretation of this derivation. **Solution:**

To prove: $Var(X) = E[X^2] - (E[X])^2$

We know that $Var(X) = E[(X - E[X])^2]$

$= E[X^2 - 2.X.E[X] + (E[X])^2]$

$= E[X^2] - 2.E[X.E[X]] + E[E[X]^2]$

$= E[X^2] - 2(E[X])^2 + (E[X])^2$

$= E[X^2] - (E[X])^2$

Hence Proved, $Var(X) = E[X^2] - (E[X])^2$

Interpretation: Variance is the concentration of a random variable around its mean.

(i) (2 points) Let Y_1 and Y_2 denote the number of symptoms detected in each of the above two patients respectively, where $Y_1, Y_2 \in [1, 2, 3, 4]$. Then calculate the following probabilities:
(i) $E[Y_1 Y_2]$;
Using the property we get,
$E[Y_1 Y_2] = E[Y_1].E[Y_2]$
By using the values from the previous question,
$E[Y_1 Y_2] = (2.5).(1.5)$
$E[Y_1 Y_2] = 3.75$

(ii) $Var[Y_1 - Y_2]$
Using the formula,
$Var[Y_1 - Y_2] = (1)^2.Var(Y_1) + (-1)^2.Var(Y_2) + 2.CoVar(Y_1 - Y_2)$
Here the CoVar is 0 because they are independent events,
$Var[Y_1 - Y_2] = Var(Y_1) + Var(Y_2)$
$Var[Y_1] = E[X^2] + (E[X])^2$
$Var[Y_1] = (1)^2.(20/80) + (2)^2.(20/80) + (3)^2.(20/80) + (4)^2.(20/80) - (2.5)^2$
$Var[Y_1] = 1.25$ ....(1)
$Var[Y_2] = (1)^2.(70/100) + (2)^2.(15/100) + (3)^2.(10/100) + (4)^2.(5/100) - (1.5)^2$
$Var[Y_2] = 0.75$ ....(2)
Adding (1) and (2)
$Var[Y_1 - Y_2] = 2$

(j) (2 points) Among a population of n people, let X be the number of people that test positive. What is the expectation of X, $E[X]$? What is the variance of X, $Var[X]$? Make sure to include all the steps in the calculation.

Solution:
Given: Let 'X' be the number of people that test positive, among the population of 'n'.
The probability of testing positive is 0.01576 ... (We know this from Q1-(d))
Expected value = E[X] = n.p,
Where n is the total population and p is the probability.
Therefore, E[X] = n.(0.01576)
For Variance = $Var[X] = n.p(1 - p)$
$Var[X] = n.(0.01576)(1 - 0.015760)$
$Var[X] = n.(0.01576)(0.98424)$
$Var[X] = n.(0.01551162240)$

(k) (2 points) Define bias error and variance error. What do you understand by Bias-Variance trade-off?

Bias Error:

Bias is the phenomenon that distorts the results of an algorithm in favor of or against it. It generally occurs when there is a skewness in the dataset. Bias errors are differences between the model's predicted values and the actual or anticipated values that occur when it makes predictions.

Variance Error:
The variance of a random variable shows how far away it is from predicted value.

Bias-Variance Trade-off:
In order to prevent over-fitting and under-fitting in the machine learning model, bias and variance must be carefully considered when the model is being built.

Problem 2 (20 points)

(a) (2 points) Show that for any arbitrary matrix $X \in \mathbb{R}_{m\times n}$, the matrix $XX^\top$ is always positive semi-definite.
Solution:
For an arbitrary matrix $X \in \mathbb{R}_{m\times n}$, U is a positive matrix if $XUX^T \geq 0 \forall U$
$U(XX^T)U^T$
Now, $(UX)(UX)^T$
$||UX||^2 \geq 0$
Therefore,UX are positive semi-definite.

(b) Recall that the SVD of a rank-r matrix M has the form

$$M = \sum_{i=1}^{r} \sigma_i u_i v_i^T,$$

where $\{u_i\}_{i=1}^r$ denote the left singular vectors, $\{v_i\}_{i=1}^r$ denote the right singular vectors, and $\{\sigma_i\}_{i=1}^r$ denote the singular values.

i) (2 points) Let

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Calculate the left and right singular vectors $\{u_i\}_{i=1}^r$ and $\{v_i\}_{i=1}^r$ of A. Then show that $\{u_i\}_{i=1}^r$ and $\{v_i\}_{i=1}^r$ are the eigenvectors of $AA^\top$ and $A^\top A$.
Solution:

See solution in python file $->$ Question-2-PS1.ipynb

From the code we can see that the eigen values are 4,2 and 0,

$$\{u_i\}_{i=1}^{r} = \begin{pmatrix} -0.7071 & 0 & -0.7071 \\ -0.7071 & 0 & 0.7071 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\{v_i\}_{i=1}^{r} = \begin{pmatrix} -0.7071 & 0.7071 \\ -0.7071 & -0.7071 \end{pmatrix}$$

$$A.A^T = \begin{pmatrix} 2 & 2 & 0 \\ 2 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

For $\{u_i\}_{i=1}^{r}$ to be an eigen vector of $A.A^T$

$A.A^T.u_i = \lambda.u_i$

$$A.A^T.u_1 = \begin{pmatrix} -2.8284 \\ -2.8284 \\ 0 \end{pmatrix} = 4.u_1$$

Similarly,

$$A.A^T.u_2 = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} = 2.u_2$$

$$A.A^T.u_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = 0.u_3$$

When we multiply u_1,u_2 and u_3 with the eigen values i.e. 4,2 and 0. The product equals with $A.A^T.u_i$.

Therefore, we can say that the $\{u_i\}_{i=1}^{r}$ is the eigenvector of $A.A^T$

Similarly we can find $\{u_i\}_{i=1}^{r}$ to be eigenvector of $A^T.A$

$A^T.A.v_i = \lambda.v_i$

$$A^T.A = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$

$$A^T.A.v_1 = \begin{pmatrix} -2.8284 \\ -2.284 \end{pmatrix} = 4.v_1$$

$$A^T.A.v_2 = \begin{pmatrix} 1.414 \\ -1.414 \end{pmatrix} = 2.v_2$$

$$A^T.A.v_3 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = 0.v_3$$

When we multiply v_1,v_2 and v_3 with the eigen values i.e. 4,2 and 0. The product equals with $A^T.A.v_i$.

Therefore, we can say that the $\{v_i\}_{i=1}^{r}$ is the eigenvector of $A^T.A$

ii) (5 points) Let $M \in \mathbb{R}^{m \times n}$ be an arbitrary real-valued rank-r matrix, show that the eigenvectors of MM^T and M^TM are $\{u_i\}_{i=1}^r$ and $\{v_i\}_{i=1}^r$ respectively.

Solution:

Showing eigen vectors in matrix form:

$\{u_i\}_{i=1}^r = U$

$\{v_i\}_{i=1}^r = V$

Using SVD,

$M = U.D.V^T$

$M^T = (U.D.V^T)^T$

$M.M^T = (U.D.V^T).(V.D^T.U^T)$

$M.M^T = U.D^2.U^T$

Multiplying both sides by U we get,

Here, U is an orthonormal matrix, so $U.U^T$ is an identity matrix

$M.M^T.U = U.D^2$

Hence, U is an eigen vector of $M.M^T$

D is a diagonal matrix, therefore D^2 will also be a diagonal and the square of each i^th eigen value of $M.M^T$

We can get V by the same method,

$M^T.M = (V.D^T.U^T).(U.D.V^T)$

$M^T.M = V.D^2.V^T$

Multiplying V on both sides we get,

$M^T.M.V = V.D^2$

Hence, V is an eigen vector of $M^T.M$

(c) Recall that the best rank-k approximation of M in Frobenius norm is attained by

$$B = \sum_{i=1}^{k} \sigma_i u_i v_i^T.$$

i) (2 points) For the matrix A defined above, calculate the best rank-1 approximation of A in Frobenius norm. Then find out the approximation error $||M - B||_F$.

Solution:

From the above solution we know that, for k=1,

$B = \sigma_i.u_i.v_1^t + \sigma_i.u_1.v_1^t + \sigma_i.u_1.v_1^t$

$B = \sigma_1.u_1.v_1^t$

$$B = 2. \begin{pmatrix} -0.7071 \\ -0.7071 \\ 0 \end{pmatrix} . \begin{pmatrix} -0.7071 & -0.7071 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{pmatrix}$$

Calculating the approx error,

$||M - B||_f = \sqrt{\sum_{i=1}^{3} \sigma_i^2}$

$= \sqrt{\sigma_2^2 + \sigma_3^2}$

$\sqrt{2 + 0}$

$= 1.414$

ii) (5 points) Let $M \in \mathbb{R}^{m \times n}$ be an arbitrary real-valued rank-r matrix. Show that

$$||M - B||_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2}.$$

Solution:

$B = \sum_{i=1}^{k} \sigma_i.u_i.v_i^t$

$||M - B||_F = || \sum_{i=1}^{\gamma} \sigma_i.u_i.v_i^t - \sum_{i=1}^{k} \sigma_i.u_i.v_i^t||$
Let, $\sigma_i.u_i.v_i = x_i$
$||M - B||_F = || \sum_{i=1}^{\gamma} x_i - \sum_{i=1}^{k} x_i||$
$||M - B||_F = ||[x_1 + x_2 + \cdots + x_\gamma] - [x_1 + x_2 + \cdots + x_f]||$
$||M - B||_F = || \sum_{i=k+1}^{\gamma} x_i||$
We know that, $||A||_F = \sqrt{trace(A^T.A)}$ ..
$||M - B||_F = \sqrt{trace((\sum_{i=k+1}^{\gamma} x_i)^T . \sum_{i=k+1}^{\gamma} x_i)}$
Re-substituting the value of $x_i = \sigma_i.u_i.v_i$
$||M - B||_F = \sqrt{trace((\sum_{i=k+1}^{\gamma} \sigma_i.u_i.v_i^t)^T . \sum_{i=k+1}^{\gamma} \sigma_i.u_i.v_i^T)}$
$||M - B||_F = \sqrt{trace(\sum_{i=k+1}^{\gamma} . \sum_{j=k+1}^{\gamma} \sigma_j \sigma_i.u_j^T.v_i u_i.v_i^T)}$
Since the vectors are orthonormal we know that,
$u_i.u_j^T = 0 if i \neq j$
$u_i.u_j^T = 1$ if i = j
$||M - B||_F = \sqrt{trace(\sum_{i=k+1}^{\gamma} . \sum_{j=k+1}^{\gamma} \sigma_i \sigma_j}$
Here σ is a constant,
Therefore, $\sqrt{trace(\sum_{i=k+1}^{\gamma} \sigma_i^2)}$
$= \sqrt{\sum_{i=k+1}^{\gamma} \sigma_i^2}$

Hence Proved, $||M - B||_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2}$

(d) (4 points) Write a Python file to verify your calculation in (b-i) and (c-i). You may find the library `numpy.linalg.svd` and `numpy.linalg.eig` useful.

See solution in python file $->$ Question-2-PS1.ipynb

Problem 3 (15 points)

(a) (6 points) For vectors $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{a} \in \mathbb{R}^n$ and matrices $\mathbf{X} \in \mathbb{R}^{n \times n}, \mathbf{A} \in \mathbb{R}^{n \times n}$, show the following:

(i) $\dfrac{\partial \boldsymbol{a}^T \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}$.

Solution:

$Let, a = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}, x = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$

$a^T.x = \begin{bmatrix} a_1 \\ a_2 \\ a_n \end{bmatrix} \times \begin{bmatrix} x_1 & x_2 & x_n \end{bmatrix} = \begin{bmatrix} a_1.x_1 & a_2.x_2 & \cdots & a_n.x_n \end{bmatrix}$

Taking partial derivative on both sides with respect to x,

$$\frac{\partial a^T x}{\partial x} = \frac{\partial [a_1.x_1 a_2.x_2 ... a_n.x_n]}{\partial x}$$

Here, a^T is constant, so the partial derivative would be

$$\frac{\partial a^T x}{\partial x} = a.\frac{\partial [x_1 x_2 ... x_n]}{\partial x}$$

Therefore,

$$\frac{\partial a^T x}{\partial x} = a.\frac{\partial x}{\partial x}$$

$$\frac{\partial a^T x}{\partial x} = a$$

(ii) $\dfrac{\partial \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}}{\partial \boldsymbol{x}} = (\boldsymbol{A} + \boldsymbol{A}^{\boldsymbol{T}})\boldsymbol{x}$.

Solution:

$$\frac{\partial x^T A x}{\partial x} = \frac{\partial x^T (Ax)}{\partial x} + \frac{\partial (x^T A) x}{\partial x}$$

Let $Ax = \begin{pmatrix} p_1 & p_2 & \cdots & p_n \end{pmatrix}^T$

$x = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}^T -> x^T = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}$

Now $y = x^T(Ax) = [x_1p_1 + x_2p_2 + \cdots + x_np_n] -> \frac{\partial y}{\partial x_\gamma}$

$\frac{\partial y}{\partial x} = \begin{pmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \cdots & \frac{\partial y}{\partial x_n} \end{pmatrix}^T$

$\frac{\partial y}{\partial x} = \begin{pmatrix} p_1 & p_2 & \cdots & p_n \end{pmatrix}^T = Ax$

$\frac{\partial x^T Ax}{\partial x} = Ax$

Let $x^T A = B^T$

$\frac{\partial B^T x}{\partial x} = B = (x^T A)^T = A^T x$

$\frac{\partial (x^T A)x}{\partial x} = A^T x$

$\frac{\partial (x^T A)x}{\partial x} = \frac{\partial x^T (Ax)}{\partial x} + \frac{\partial (x^T A)x}{\partial x} = Ax + A^T x = (A + A^T)x$

Now, we can get from the above question 3.a.ii,

$\frac{\partial 2.y^T.A.x}{\partial x} = (A^T.A + A^T.A)x$

(iii) $\dfrac{\partial ||\boldsymbol{y} - \boldsymbol{Ax}||_2^2}{\partial \boldsymbol{x}} = 2\boldsymbol{A}^T(\boldsymbol{Ax} - \boldsymbol{y})$.

Solution:

$||y - Ax||_2^2 = \sum_{i=1}^n (y - (Ax)_i)^2$

$||y - Ax||_2^2 = \sum_{i=1}^n [y_i^2 - 2.y_i.(Ax)_i + (Ax)_i^2]$

Converting it into matrix form we get,

$= y.y^T - 2.y^T.Ax + x^T.A^T.Ax$

Taking partial derivatives,

$\frac{\partial y.y^T - 2.y^T.Ax + x^T.A^T.Ax}{\partial x} = \frac{\partial y.y^T}{\partial x} - \frac{2.y^T.Ax}{\partial x} + \frac{\partial x^T.A^T.Ax}{\partial x}$

Since we are taking the derivative with respect to x, y is a constant therefore it will be zero.

Now, we can get from the above question 3.a.ii,

$\frac{\partial 2.y^T.A.x}{\partial x} = (2.A^T.y)$

$\frac{\partial x^T.A^T.Ax}{\partial x} = (A^T.A + A^T.A)x$

$\frac{||y - Ax||_2^2}{x} = -(2.A^T.y) + (A^T.A + A^T.A)x$

$\frac{||y-Ax||_2^2}{x} = -2.A^T.y + 2.A^T.Ax$
$\frac{||y-Ax||_2^2}{x} = 2.A^T(Ax - y)$
Hence, Proved.

(b) (4 points) You are given a training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, where $\boldsymbol{x_i} \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Consider the regression problem

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i))^2.$$

What is the minimizer of the above regression problem? Provide all steps of your derivation. Feel free to assume that the rank of $\{\boldsymbol{x}_i\}_{i=1}^n$ is equal to d.]

Solution:

To minimize θ partial derivation need to be minimum i.e 0,
$\frac{\partial \min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i))^2}{\partial \theta} = 0$
Converting the above equation into matrix form we get,
$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i))^2 = 1/N(y - X.\theta)^T(y - X.\theta)$
$= 1/N(y^T - \theta^T.X^T)(y - X.\theta)$
Now, taking derivative with respect to θ

$\frac{\partial [1/N(y^T - \theta^T.X^T)(y - X.\theta)]}{\partial \theta}$

$1/N \frac{\partial y^T.y - y^T.X.\theta - \theta^T.X^T.y + \theta^T.X^T.X.\theta}{\partial \theta}$

Using the above proof of 3.a ,

$= 1/N(0 - X^T.y - X^T.y + 2.X^T.X.\theta)$

Minimizing the equation to 0,

$-2.X^T.y + 2.X^T.X.\theta = 0$
$-2(X^T.y - X^T.X.\theta) = 0$
$X^T.y - X^T.X.\theta = 0$
$Therefore, \theta = (X^T.X)^{-1}X^T.y$

(c) (5 points) Let the cost function to minimize is:

$$J(w) = \sum_{i=1}^{n} (y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i))^2 + \lambda \sum_{j=0}^{d} {\theta_j}^2$$

Prove that the vector $w^\star$ that minimizes $J(w)$ is:

$$w^\star = (X^\top X + \lambda I)^{-1} X^\top y,$$

where X is the n by d design matrix, whose i-th row is x_i, and $y = (y_1, ..., y_n)^\top$.

Solution:

Converting the above equation into matrix form we get,

$(y - X.\theta)^T(y - X.\theta) + \lambda.\theta^T.\theta$

$(y^T - \theta^T.X^T)(y - X.\theta) + \lambda.\theta^T.\theta$

Now taking partial derivative with respect to θ,

$\frac{\partial[(y^T - \theta^T.X^T)(y - X.\theta)]}{\partial\theta} + \frac{\partial(\lambda.\theta^T.\theta)}{\partial\theta}$

From the above question 3.b we know that,

$\frac{\partial[(y^T - \theta^T.X^T)(y - X.\theta)]}{\partial\theta} = -2(X^T.y - X^T.X.\theta)$

Now for the second term,

$\frac{\partial(\lambda.\theta^T.\theta)}{\partial\theta} = \frac{\partial(\lambda.\theta^2)}{\partial\theta}$

$= 2.\lambda.I.\theta$

as $\theta = I.\theta$

$-2(X^T.y - X^T.X.\theta) + 2.\lambda.I.\theta = 0$

$-2(X^T.y - X^T.X.\theta + \lambda.I.\theta) = 0$

$X^T.y - X^T.X.\theta - \lambda.I.\theta = 0$

$X^T.X.\theta - \lambda.I.\theta = X^T.y$

$(X^T.X - \lambda.I)\theta = X^T.y$

$\theta = (X^T.X - \lambda.I)^{-1}X^T.y$

Hence the minimizer $w*$ is $(X^T.X - \lambda.I)^{-1}X^T.y$

Problem 4 (15 points)

We consider a regression problem for predicting the demand of bike-sharing services in Washington D.C.[1] The prediction task is to predict the demand for the bikes (column cnt) given the other features: ignore the columns instant and dteday. Use the day.csv file from the data folder.

[1] https://www.kaggle.com/datasets/marklvl/bike-sharing-dataset?search=bike+demand+Washington&select=Readme.txt. You can also find a Readme.txt file that explains all the features in the dataset.

(a) (4 points) Write a Python file to load day.csv.[2] Compute the correlation coefficient of each feature with the response (i.e., cnt). Include a table with the correlation coefficient of each feature with the response. Which features are positively correlated (i.e., have positive correlation coefficient) with the response? Which feature has the highest positive correlation with the response?

(b) (2 points) Were you able to find any features with a negative correlation coefficient with the response? If not, can you think of a feature that is not provided in the dataset but may have a negative correlation coefficient with the response?

(c) (5 points) Now, divide the data into training and test sets with the training set having about 70 percent of the data. Import `train_test_split` from `sklearn` to perform this operation. Use an existing package to train a multiple linear regression model on the training set using all the features (except the ones excluded above). Report the coefficients of the linear regression models and the following metrics on the training data: (1) RMSE metric; (2) R^2 metric.

[Hint: You may find the libraries `sklearn.linear_model.LinearRegression` useful.]

(d) (2 points) Next, use the test set that was generated in the earlier step. Evaluate the trained model in step (c) on the testing set. Report the RMSE and R^2 metrics on the testing set.

(e) (2 points) Interpret the results in your own words. Which features contribute mostly to the linear regression model? Is the model fitting the data well? How large is the model error?

Problem 5 (10 points)

This question should be answered using the `Diabetes` data set that is readily available in the `Scikit-learn` library.[3] This data set has information about 442 patients and whether they have suffered from diabetes or not.

(a) (2 points) Fit a multiple regression model to predict `Diabetes` using `Age`, `Sex`, `BMI`, and `BP`.

(b) (4 points) Provide an interpretation of each coefficient in the model. Be careful—some of the variables in the model are qualitative!

(c) (2 points) Write out the model in equation form, being careful to handle the qualitative variables properly.

[2] Refer to `https://docs.python.org/3/library/csv.html` on how to load a csv file in Python.

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html`. You can find the description of this data set at `https://scikit-learn.org/stable/datasets/toy_dataset.html`.

(d) (2 points) Using the model from (c), obtain 95% confidence intervals for the coefficient(s).

Problem 6 (20 points)

We will now perform cross-validation on a simulated data set.

(a) (2 points) Generate a simulated data set as follows:

```
numpy.random.seed(12345)
x = numpy.random.normal(0, 1, (200))
y = x + 2 * x**2 - 2 * x**3 + numpy.random.normal(0, 1, (200))
```

In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

Solution:

Here n represents the size of the dataset which is 200 and p refers to the dimension of each data point which is 1.

(b) (2 points) Create a scatterplot of X against Y. Comment on what you find. (Hint: You may find `matplotlib.pyplot.plot()` helpful)

Soltion:

The scatterplot of X and Y plots a cubical graph

(c) (9 points) Set a random seed `123`, and then compute the leave-one-out cross validation errors that result from fitting the following five models using least squares:

(i) $Y = \beta_0 + \beta_1 X + \varepsilon$

(ii) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \varepsilon$

(iii) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \varepsilon$

(iv) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \varepsilon$

(v) $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \beta_5 X^5 + \varepsilon$

[Hint: You may find `LeaveOneOut()` and `cross_val_score()` in `sklearn.model_selection` helpful.]

(d) (2 points) Repeat (c) using another random seed `12345`, and report your results. Are your results the same as what you got in (c)? Why?

Solution:

Since we are splitting the dataset into n-1 segments,(n is the total number of observations) the order in which the parts appear doesn't affect the outcome using seeds 123 and 12345. Because of this, the outcomes are unaffected by the seed.

(e) (5 points) Which of the models in (c) had the smallest leave-one-out cross validation error? Is this what you expected? Explain your answer.

Solution:

The model with a 3rd degree polynomial produced the lowest cross validation score. Given that the relation between x and y is a the same 3rd degree polynomial, I anticipated that the model would provide the highest score.

Question 2.b).i)

Note: The values of the range e^-15 should be considered as 0

In [59]:
```
import numpy as np
```

In [60]:
```
X = np.array([[1,1],[1,1],[1,-1]])
U,D,Vt = np.linalg.svd(X)
```

Using the svd function, we get values of the matrix U, D (diagonal matrix), V^T

In [61]:
```
display("U=", U)
```

```
'U='
array([[-7.07106781e-01, -2.22044605e-16, -7.07106781e-01],
       [-7.07106781e-01, -1.11022302e-16,  7.07106781e-01],
       [-5.55111512e-17,  1.00000000e+00, -2.25745422e-16]])
```

In [62]:
```
display("D=",D)
```

```
'D='
array([2.        , 1.41421356])
```

In [63]:
```
display("V^T",Vt)
```

```
'V^T'
array([[-0.70710678, -0.70710678],
       [ 0.70710678, -0.70710678]])
```

In [64]:
```
X = np.dot(U[:,:2]* D ,Vt)
```

In [65]:
```
A = np.dot(X , X.T)
display("A is the matrix multiplication of X.X^T",A)
```

```
'A is the matrix multiplication of X.X^T'
array([[ 2.00000000e+00,  2.00000000e+00, -3.33066907e-16],
       [ 2.00000000e+00,  2.00000000e+00, -1.11022302e-16],
       [-3.33066907e-16, -1.11022302e-16,  2.00000000e+00]])
```

In [66]:
```
display("The eigen value are:")
np.linalg.eig(A)
```

Out[66]:
```
'The eigen value are:'
(array([4.00000000e+00, 2.22044605e-16, 2.00000000e+00]),
 array([[ 7.07106781e-01,  7.07106781e-01,  5.55111512e-17],
        [ 7.07106781e-01, -7.07106781e-01,  1.66533454e-16],
        [-1.17756934e-16,  1.17756934e-16,  1.00000000e+00]]))
```

Question 2.d)

Question.2.c).i)

In [85]:
```
M = np.array([[1,1],[1,1],[1,-1]])

B = 2*U[:,0].reshape(3,1)@Vt[0].reshape(1,2)


display("B",B)

print("Frobenius Norm of (M-B)=",np.linalg.norm(M-B,"fro"))
```

```
'B'
array([[1.00000000e+00, 1.00000000e+00],
       [1.00000000e+00, 1.00000000e+00],
       [7.85046229e-17, 7.85046229e-17]])
Frobenius Norm of (M-B)= 1.414213562373095
```

Question.2.b).i)

In [86]:
```
A = np.array([[1,1],[1,1],[1,-1]])

U,D,Vt = np.linalg.svd(A, full_matrices = True)

display("Left singular matrix")
print(U)
display("Right singular matrix")
print(Vt.T)
```

```
'Left singular matrix'
[[-7.07106781e-01 -2.22044605e-16 -7.07106781e-01]
 [-7.07106781e-01 -1.11022302e-16  7.07106781e-01]
 [-5.55111512e-17  1.00000000e+00 -2.25745422e-16]]
'Right singular matrix'
[[-0.70710678  0.70710678]
 [-0.70710678 -0.70710678]]
```

In [87]:
```
D = np.append(D,0)
```

In [88]:
```
display("Eigen values of:")
print(D**2)
```

```
'Eigen values of:'
[4. 2. 0.]
```

In [89]:
```
AAt = A@A.T
display("A.A^t")
print(AAt)

AtA = A.T@A
display("AtA")
print(AtA)
```

```
'A.A^t'
[[2 2 0]
 [2 2 0]
 [0 0 2]]
'AtA'
[[3 1]
 [1 3]]
```

In [90]:
```
for i in range(U.shape[1]):
    print("A.At.U[" + str(i) + "]=",AAt@U[:,i])

for i in range(len(D)):
    print("sigma[" + str(i) + "]^2.U[" + str(i) + "]=" ,(D[i]**2)*U[:,1])
```

```
A.At.U[0]= [-2.82842712e+00 -2.82842712e+00 -1.11022302e-16]
A.At.U[1]= [-6.66133815e-16 -6.66133815e-16  2.00000000e+00]
A.At.U[2]= [ 4.44089210e-16  4.44089210e-16 -4.51490845e-16]
sigma[0]^2.U[0]= [-8.8817842e-16 -4.4408921e-16  4.0000000e+00]
sigma[1]^2.U[1]= [-4.44089210e-16 -2.22044605e-16  2.00000000e+00]
sigma[2]^2.U[2]= [-0. -0.  0.]
```

In [91]:
```
for i in range(Vt.T.shape[1]):
    print("At.A.V[" + str(i) + "]=",AtA@Vt.T[:,i])

for i in range(len(D)):
    print("sigma[" + str(i) + "]^2.V[" + str(i) + "]=" ,(D[i]**2)*Vt.T[:,1])
```

```
At.A.V[0]= [-2.82842712 -2.82842712]
At.A.V[1]= [ 1.41421356 -1.41421356]
sigma[0]^2.V[0]= [ 2.82842712 -2.82842712]
sigma[1]^2.V[1]= [ 1.41421356 -1.41421356]
sigma[2]^2.V[2]= [ 0. -0.]
```

In [432]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/dock
# For example, here's several helpful packages to load

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

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list a

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets
# You can also write temporary files to /kaggle/temp/, but they won't be saved ou
```

```
/kaggle/input/bike-sharing-dataset/hour.csv
/kaggle/input/bike-sharing-dataset/Readme.txt
/kaggle/input/bike-sharing-dataset/day.csv
/kaggle/input/bike-sharing-dataset/bike-sharing-dataset/hour.csv
/kaggle/input/bike-sharing-dataset/bike-sharing-dataset/Readme.txt
/kaggle/input/bike-sharing-dataset/bike-sharing-dataset/day.csv
```

In [433]:
```
from numpy import argmax
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
```

In [434]:
```
df = pd.read_csv("../input/bike-sharing-dataset/day.csv")
```

In [435]:
```
df.head()
```

Out[435]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	aten
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.3636
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.3537
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.1894
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.2121
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.2292

In [436]:
```
df.columns
```

Out[436]:
```
Index(['instant', 'dteday', 'season', 'yr', 'mnth', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
       'casual', 'registered', 'cnt'],
      dtype='object')
```

In [437]:
```
df.drop(['instant', 'dteday','casual','registered'], inplace=True ,axis=1)
```

Question 4. a)

In [438]:
```
correlations = df.corrwith(df['cnt'])
print(correlations)
```

```
season        0.406100
yr            0.566710
mnth          0.279977
holiday      -0.068348
weekday       0.067443
workingday    0.061156
weathersit   -0.297391
temp          0.627494
atemp         0.631066
hum          -0.100659
windspeed    -0.234545
cnt           1.000000
dtype: float64
```

Features with positive correlation are:

- **season**
- **yr**: year
- **mnth**: mmonths
- **weekday**: days of the week
- **workingday**: whether the day is neither a weekend nor holiday
- **temp**: temperature in Celsius
- **atemp**: "feels like" temperature in Celsius

The feature with highest correlation is **'atemp'**.

Question 4. b)

Features with negative correlation are:

- **holiday**: whether the day is considered a holiday
- **weathersit**
- **hum**: humidity
- **windspeed**

In [439]:
```
df.head()
```

Out[439]:

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	wi
0	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0
1	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0
2	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0
3	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0
4	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0

Question 4.c)

In [440]:
```
import category_encoders as ce
```

In [441]:
```
encoder = ce.OneHotEncoder(cols=('season','yr','mnth','holiday','weekday','weathe
```

In [442]:
```
from sklearn.model_selection import train_test_split
```

In [443]:
```
encoded = encoder.fit_transform(df)
new_df = pd.concat([df.drop(['season','yr','mnth','holiday','weekday','weathersit
```

In [444]:
```
new_df.head()
```

Out[444]:

	workingday	temp	atemp	hum	windspeed	season_1.0	season_2.0	season_3.0	sea
0	0	0.344167	0.363625	0.805833	0.160446	1	0	0	
1	0	0.363478	0.353739	0.696087	0.248539	1	0	0	
2	1	0.196364	0.189405	0.437273	0.248309	1	0	0	
3	1	0.200000	0.212122	0.590435	0.160296	1	0	0	
4	1	0.226957	0.229270	0.436957	0.186900	1	0	0	

5 rows × 41 columns

In [445]:
```
X = new_df.drop(['cnt','workingday'], axis = 1)
y = new_df['cnt']
```

Workingday is dropped because it is related with holiday

In [446]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_s
```

In [447]:
```
from sklearn.linear_model import LinearRegression
```

In [448]:
```
LR = LinearRegression()
```

In [449]:
```
LR.fit(X_train, y_train)
```

Out[449]:
```
LinearRegression()
```

In [450]:
```
LR.coef_
```

Out[450]:
```
array([-1.22890549e+16,  1.84050000e+03, -6.52083811e+02, -1.10960252e+03,
       -9.46236603e+02,  1.10402434e+02,  1.16859425e+02,  7.86333604e+02,
       -9.91436156e+02,  9.71052084e+02, -7.61286502e+01,  5.92396907e+01,
        4.90914051e+02,  8.37791449e+01,  3.26381701e+02,  1.47548247e+01,
       -5.47806058e+02, -2.31291306e+02,  5.58128836e+02,  1.83645124e+02,
       -4.26070831e+02, -3.40710852e+02,  2.85023358e+02, -2.90367108e+02,
        1.17002260e+02, -3.42486123e+02, -5.21637419e+01, -4.73031381e+01,
        6.56444483e+01,  6.77283970e+01,  1.36528345e+02,  2.31578465e+02,
        7.52826270e+02, -1.04573438e+03,  1.22890549e+16,  1.90175000e+03,
       -6.93265625e+02, -1.15386719e+03])
```

In [451]:
```
pred = LR.predict(X_train)
```

In [452]:
```
from sklearn.metrics import mean_squared_error,r2_score
```

Question 4. c)

The rmse and r^2 value for training dataset using Linear Regression and Random Forest are:

In [453]:
```
mean_squared_error(y_train, pred,squared=False)
```

Out[453]:
```
759.0600752359645
```

In [454]:
```
r2_score(y_train, pred)
```

Out[454]:
```
0.8416632435493496
```

In [455]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [456]:
```
RF = RandomForestRegressor()
```

In [457]:
```
RF.fit(X_train,y_train)
```

Out[457]:
```
RandomForestRegressor()
```

In [458]:
```
mean_squared_error(y_train, RF.predict(X_train),squared=False)
```

Out[458]:
```
270.052348153192
```

In [459]:
```
r2_score(y_train,RF.predict(X_train))
```

Out[459]:
```
0.9799587243418628
```

In [460]:
```
y_pred = LR.predict(X_test)
```

Question 4. d)

The rmse and r^2 value for test using Linear Regression and Random Forest are:

In [461]:
```
mean_squared_error(y_test, y_pred,squared=False)
```

Out[461]:
```
774.1783135397773
```

In [462]:
```
r2_score(y_test,y_pred)
```

Out[462]:
```
0.8493698992205666
```

In [463]:
```
RF.fit(X_test,y_test)
```

Out[463]:
```
RandomForestRegressor()
```

In [464]:
```
mean_squared_error(y_test, RF.predict(X_test),squared=False)
```

Out[464]:
```
293.62906371738046
```

In [465]:
```
r2_score(y_test,RF.predict(X_test))
```

Out[465]:
```
0.9783315501077117
```

Question 4. e)

atemp has the highest positive correlation and windspeed has the highest negative correlation, these are the features which give significant contribution.

The model has a good fit with r square value of 0.979 while using Random Forest and 0.849 using Linear Regression.

The same goes for rmse value.

This indicates that Random Forest has a better fit as compared to Linear Regression for this model.

The model fits the data well using Linear Regression which answers to the question how well the model fits.

In [125]:
```
import numpy as np
import pandas as pd
from sklearn import linear_model
from sklearn.datasets import load_diabetes
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
```

In [126]:
```
diabetes_X, diabetes_y = load_diabetes(return_X_y=True,as_frame=True )
```

In [127]:
```
diabetes_X = diabetes_X[['age','sex','bmi','bp']]
```

In [128]:
```
diabetes_X
```

Out[128]:

	age	sex	bmi	bp
0	0.038076	0.050680	0.061696	0.021872
1	-0.001882	-0.044642	-0.051474	-0.026328
2	0.085299	0.050680	0.044451	-0.005671
3	-0.089063	-0.044642	-0.011595	-0.036656
4	0.005383	-0.044642	-0.036385	0.021872
...	...	...	...	...
437	0.041708	0.050680	0.019662	0.059744
438	-0.005515	0.050680	-0.015906	-0.067642
439	0.041708	0.050680	-0.015906	0.017282
440	-0.045472	-0.044642	0.039062	0.001215
441	-0.045472	-0.044642	-0.073030	-0.081414

442 rows × 4 columns

In [129]:
```
diabetes_X[["sex"]].drop_duplicates()
```

Out[129]:

	sex
0	0.050680
1	-0.044642

Here "sex" is categorical, so converting it to 0 & 1

In [130]:
```
diabetes_X["sex"] = diabetes_X["sex"].replace(np.unique(diabetes_X["sex"])[0],0)
diabetes_X["sex"] = diabetes_X["sex"].replace(np.unique(diabetes_X["sex"])[1],1)

diabetes_X[["sex"]].drop_duplicates()
```

Out[130]:

	sex
0	1.0
1	0.0

In [131]:
```
X_train, X_test, y_train, y_test = train_test_split(diabetes_X, diabetes_y, test_
```

In [132]:
```
X_test = sm.add_constant(X_test)
X_train = sm.add_constant(X_train)
```

In [133]:
```
model = sm.OLS(y_train, X_train).fit()
```

In [134]:
```
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 target   R-squared:                       0.424
Model:                            OLS   Adj. R-squared:                  0.417
Method:                 Least Squares   F-statistic:                     55.98
Date:                Fri, 30 Sep 2022   Prob (F-statistic):           2.40e-35
Time:                        20:47:00   Log-Likelihood:                -1701.0
No. Observations:                 309   AIC:                             3412.
Df Residuals:                     304   BIC:                             3431.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        155.7696      4.734     32.904      0.000     146.454     165.085
age           49.8484     78.139      0.638      0.524    -103.913     203.610
sex          -10.2272      6.962     -1.469      0.143     -23.928       3.473
bmi          824.5298     77.285     10.669      0.000     672.450     976.610
bp           416.2953     80.647      5.162      0.000     257.598     574.993
==============================================================================
Omnibus:                        8.895   Durbin-Watson:                   1.705
Prob(Omnibus):                  0.012   Jarque-Bera (JB):                5.327
Skew:                           0.138   Prob(JB):                       0.0697
Kurtosis:                       2.419   Cond. No.                         31.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

Question 5. b)

The coefficients are:

In [135]:
```
x = pd.DataFrame(model.params).T
x
```

Out[135]:

	const	age	sex	bmi	bp
0	155.76959	49.848355	-10.227191	824.529801	416.295292

All the variables are normalized, except for 'sex' since it is already converted.

age: When the 'age' increases by 1,the predicted value increases by 49.85, all other variables are constant.

sex: 'sex' is categorical, when 'sex'=0 there are no changes. But, when 'sex'=1, the predicted value changes by 10.23.

bmi: For every value increase in 'bmi', the predicted value increases by 824.53, all other variables are constant.

bp: When the 'age' increases by 1 the predicted value increases by 416.29, all other variables are constant.

Question 5. c)

$$Y_{pred} = \beta_0 + \beta_1 . X_1 + \beta_2 . X_2 + \beta_3 . X_3 + \beta_4 . X_4$$

$$Y_{pred} = 155.77 + (49.85 * age) - (10.23 * sex) + (824.53 * bmi) + (416.29 * bp)$$

Question 5. d)

In [136]:
```
model_con = model.conf_int(alpha=0.05).T
```

In [137]:
```
for col in model_con.columns:
    print("The 95% confidence model of " + col + " is (" + str(round(model_con[cc
```

```
The 95% confidence model of const is (146.454,165.085)
The 95% confidence model of age is (-103.913,203.61)
The 95% confidence model of sex is (-23.928,3.473)
The 95% confidence model of bmi is (672.45,976.61)
The 95% confidence model of bp is (257.598,574.993)
```

In [102]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from numpy.polynomial import Polynomial
```

In [103]:
```
np.random.seed(12345)
x = np.random.normal(0,1,(200))
y = x + 2 * x**2 - 2 * x**3 + np.random.normal(0,1,(200))
```

Type *Markdown* and LaTeX: α^2

Question 6. a)

Here n represents the size of the dataset which is 200 and p refers to the dimension of each data point which is 1.

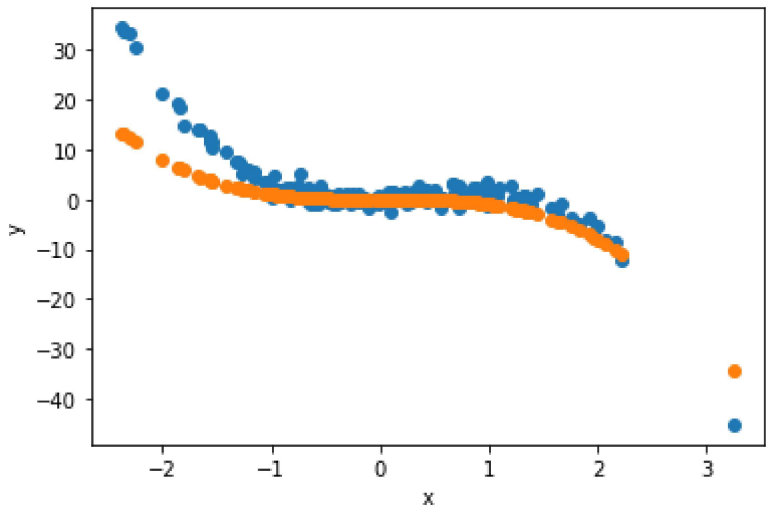
The model equation is: $y = 2x + 2x^2 - 2x^3$ As we have set the random seed to (12345), we can assert that the value of np.random.normal() will be same throughout the cell hence x + np.random.normal(0, 1, (200)) = 2x

Question 6.b)

The scatterplot between x and y generates cubical graph. There is a cubical relation.

In [104]:
```
plt.scatter(x,y)
plt.xlabel("x")
plt.ylabel("y")
plt.scatter(x,-x**3)
```

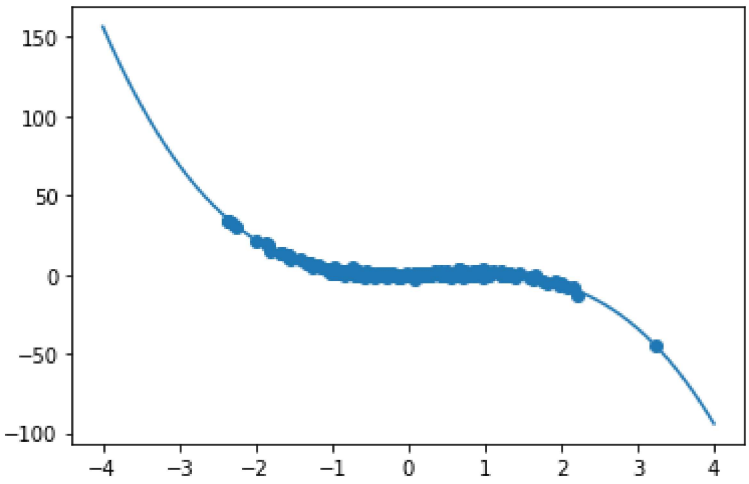
Out[104]:
```
<matplotlib.collections.PathCollection at 0x7f4cb3928210>
```



The blue indicates the plot of the generated graph and the orange indicates the cubical graph. Which shows that the similarity between them.

Type *Markdown* and LaTeX: α^2

In [105]:
```
polyline = np.linspace(-4,4,200)
model = np.poly1d(np.polyfit(x,y,3))
plt.scatter(x,y)
plt.plot(polyline, model(polyline))
plt.show()
```



In [106]:
```
print(model)
```

```
         3         2
-2.021 x + 1.957 x + 1.003 x + 0.01806
```

Question 6. c)

In [107]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import cross_val_score
import random
```

In [108]:
```
random.seed(123)
c_v_d_123 = {}

for poly_val in range(1,6):
    poly_f = PolynomialFeatures(degree = poly_val, include_bias = True)
    x_poly = poly_f.fit_transform(x.reshape(-1,1))

    model = LinearRegression()

    cv = LeaveOneOut()

    c_v_d_123[poly_val] = cross_val_score(model, x_poly, y, scoring='neg_mean_squ
```

Type *Markdown* and LaTeX: α^2

In [109]:
```
for i in range(1,6):
    print("Leave one out cross validation of degree " + str(i) + " is ",abs(c_v_c
```

```
Leave one out cross validation of degree 1 is  26.534996047663075
Leave one out cross validation of degree 2 is  32.53855587022528
Leave one out cross validation of degree 3 is  1.1079418003348256
Leave one out cross validation of degree 4 is  1.3234306760123546
Leave one out cross validation of degree 5 is  1.1099273761939197
```

In [110]:
```
x_poly
```

Out[110]:
```
array([[ 1.00000000e+00, -2.04707659e-01,  4.19052259e-02,
        -8.57832070e-03,  1.75604795e-03, -3.59476467e-04],
       [ 1.00000000e+00,  4.78943338e-01,  2.29386721e-01,
         1.09863242e-01,  5.26182678e-02,  2.52011688e-02],
       [ 1.00000000e+00, -5.19438715e-01,  2.69816579e-01,
        -1.40153177e-01,  7.28009861e-02, -3.78156507e-02],
       ...,
       [ 1.00000000e+00, -1.41341604e+00,  1.99774490e+00,
        -2.82364468e+00,  3.99098468e+00, -5.64092176e+00],
       [ 1.00000000e+00,  1.29660784e+00,  1.68119190e+00,
         2.17984660e+00,  2.82640619e+00,  3.66474043e+00],
       [ 1.00000000e+00,  2.52275209e-01,  6.36427810e-02,
         1.60554959e-02,  4.05040358e-03,  1.02181641e-03]])
```

In [111]:
```
df = pd.DataFrame(x_poly) # add the columns as x,x^2,x^3
df.head()
```

Out[111]:

	0	1	2	3	4	5
0	1.0	-0.204708	0.041905	-0.008578	0.001756	-0.000359
1	1.0	0.478943	0.229387	0.109863	0.052618	0.025201
2	1.0	-0.519439	0.269817	-0.140153	0.072801	-0.037816
3	1.0	-0.555730	0.308836	-0.171630	0.095380	-0.053005
4	1.0	1.965781	3.864293	7.596353	14.932762	29.354534

In [112]:
```
df.columns = ['β', 'x', 'x^2','x^3','x^4','x^5']
```

In [113]:
```
df.head()
```

Out[113]:

	β	x	x^2	x^3	x^4	x^5
0	1.0	-0.204708	0.041905	-0.008578	0.001756	-0.000359
1	1.0	0.478943	0.229387	0.109863	0.052618	0.025201
2	1.0	-0.519439	0.269817	-0.140153	0.072801	-0.037816
3	1.0	-0.555730	0.308836	-0.171630	0.095380	-0.053005
4	1.0	1.965781	3.864293	7.596353	14.932762	29.354534

Question 6. d)

In [114]:
```
random.seed(12345)
c_v_d_12345 = {}

for poly_val in range(1,6):
    poly_f = PolynomialFeatures(degree = poly_val, include_bias = True)
    x_poly = poly_f.fit_transform(x.reshape(-1,1))

    model = LinearRegression().fit(x_poly, y)

    cv = LeaveOneOut()

    c_v_d_12345[poly_val] = cross_val_score(model, x_poly, y, scoring='neg_mean_s
```

In [115]:
```
for i in range(1,6):
    print("Leave one out cross validation of degree " + str(i) + " is ",abs(c_v_d
```

```
Leave one out cross validation of degree 1 is  26.534996047663075
Leave one out cross validation of degree 2 is  32.53855587022528
Leave one out cross validation of degree 3 is  1.1079418003348256
Leave one out cross validation of degree 4 is  1.3234306760123546
Leave one out cross validation of degree 5 is  1.1099273761939197
```

Since we are splitting the dataset into n-1 (n is the total number of observations) segments for each step, the order in which the parts appear doesn't affect the outcome using seeds 123 and 12345. Because of this, the outcomes are unaffected by the seed.

Type *Markdown* and LaTeX: α^2

Question 6. e)

The model with a 3rd degree polynomial produced the lowest cross validation error. Given that the relation between x and y is a the same 3rd degree polynomial, Yes, I anticipated that the model would provide the best score.

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \varepsilon$$

Up to a certain point for a third degree polynomial, the bias and cross validation score reduces as the degree and complexity of the model increases. After this point, the model's complexity rises, which causes bias and cross validation to increase.