First and Last Name: Emmanuel Ifada
Student ID: 260955200

Do not write in the table

Question	1	2	total/100
Point			

Question 1 (Bayesian Regression, 60 points)

1) Show $\mathbf{X}^\top\mathbf{X}$ is non-negative definite.

Soln: Recall that X^TX is a positive semi-definite (PSD) for any matrix

$$X \in R^{\text{mxn}}$$

This means that for any $\lambda > 0$, the matrix $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ is a positive definite (no semi).
To show this:
C is an eigenvalue of $\mathbf{X}^T\mathbf{X}$ if and only if $C + \lambda$ is an eigenvalue of $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$
$\mathbf{X}^T\mathbf{X}$ being PSD implies that all its eigenvalues are non-negative. Hence, this in turn implies that all the eigenvalues of $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ are positive i.e it is positive definite.
If we also say that,

$$\mathbf{X}^\top\mathbf{X} = \lambda v$$

Assuming that v is an eigenvector of $\mathbf{X}^\top\mathbf{X}$ corresponding to eigen λ. Then

$$\mathbf{X}^\top\mathbf{X}v\ v^\top = \lambda v^\top\ v$$
$$(\mathbf{X}v)^\top(\mathbf{X}v) = \lambda v^\top v$$

thus,

$$\lambda = \frac{||\mathbf{X}_v||^2}{||v||^2} \geq 0$$

In conclusion, this means that for any $\lambda > 0$, the matrix $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ is invertible and this is what makes the ridge regression work!

2) Suppose λ_i's are the eigenvalues of $\mathbf{A}$. Show the eigenvalues of $\mathbf{A}_\lambda = \mathbf{A} + \lambda\mathbf{I}$ are $\lambda_i + \lambda$. Which values of λ guarantees $\mathbf{A}_\lambda$ to be non-negative definite?

Soln: Suppose corresponding eigenvector of eigenvalue λi is v_i, then we have $Av_i = i$.
Thus, we have

$$\begin{aligned}\mathbf{A}_\lambda v_i &= (\mathbf{A} + \lambda\mathbf{I})v_i \\ &= \mathbf{A}v_i + \lambda\mathbf{I}v_i \\ &= \lambda_i v + \lambda v \\ &= (\lambda_i + \lambda)v\end{aligned}$$

From the above, we can deduce that the eigenvalue of $\mathbf{A}_\lambda$ is $\lambda_i + \lambda$.

Furthermore, to let $\mathbf{A}_\lambda$ be non-negative definite, then all its eigenvalues must be non-negative. This means that from the equation obtained above, λ must be greater than or equals to $-\lambda_i$ i.e. $\lambda \geq -\lambda_i$ such that $\lambda - \lambda_i \geq 0$.
3) Consider the Bayesian regression

$$\begin{aligned}\mathbf{y} \mid \mathbf{X}, \mathbf{w}, \sigma^2 &\sim \mathrm{N}(\mathbf{X}\mathbf{w}, \sigma^2\mathbf{I}) \\ \mathbf{w} \mid \lambda &\sim \mathrm{N}(\mathbf{0}, \lambda\mathbf{I}) \\ \sigma, \lambda > 0 &\end{aligned}$$

and where $\mathbf{y}$ and $\mathbf{X}$ are observed data, the response and the feature matrix respectively.

4) Show the posterior distribution of $\mathbf{w} \mid \mathbf{X}, \mathbf{y}, \lambda, \sigma^2$ is multivariate Gaussian and specify its mean vector and variance-covariance matrix.

Soln: Let's consider the case of a Gaussian prior distribution on w, We can generating a dataset of $\mathcal{D} = f(x_1, y_1), ..., (x_n, y_n)$ under this model. Note that we assume that $x_1, ..., x_n$ are given, and we are generating corresponding random values for $y_1, ..., y_n$.

Having vector $\mathbf{y} \in \mathbb{R}^n$ and covariates matrix $\mathbf{X} \in R^{nXd}$, the i^{th} row of $\mathbf{y}$ and $\mathbf{X}$ corresponds to the i^th observation (y_i, x_i). It will be convenient to rewrite this model more compactly, using random vectors. For the data given above, $\mathcal{D} = (x_1, y_1), ..., (x_n, y_n)$, let $y = (y_1, ..., y_n)$ and we assume that σ^2 and λ are known.
As noted above, this amounts to a specific conditional distribution for $\mathbf{y}|\mathbf{X}$.
Since $\mathbf{w}$ is a continuous-valued random variable in $\mathbb{R}^b$, Bayes rule says that the posterior distribution of $\mathbf{w}$ given $\mathbf{y}$ and $\mathbf{X}$ is:

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{w}, \mathbf{X})p(\mathbf{w})}{\int_{\mathbb{R}^d} p(\mathbf{y}|\mathbf{w}, \mathbf{X})p(\mathbf{w})}$$

It can be re written as

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$$

Thus, using Bayesian Linear regression, we can update the posterior distribution $p(\mathbf{w}|\mathbf{y}, \mathbf{X})$ analytically.
Using proportionality:

$$p(\mathbf{w}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{w})p(\mathbf{w})$$

This means that for every dataset $\mathcal{D}$, we have a proportionality constant k such that $p(\mathbf{w}|\mathcal{D}) = kp(\mathcal{D}|\mathbf{w})p(\mathbf{w})$. Put another way, there is a function k(D) such that

$$p(\mathbf{w}|\mathcal{D}) = k\mathcal{D}p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) \quad \forall \mathbf{w}, \mathcal{D}.$$

Therefore,

$$\begin{aligned} p(\mathbf{w}|\mathcal{D}) &\propto p(\mathbf{y}|\mathbf{w}, \mathbf{X}) \\ &\propto \left[e^{-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w})}\right]\left[e^{-\frac{\lambda}{2}(\mathbf{w})^T\mathbf{w}}\right] \\ &\propto e^{-\frac{1}{2}\mathbf{w}^T(\lambda I+\sigma^{-2}\mathbf{X}^T\mathbf{X})\mathbf{w}-2\sigma^{-2}\mathbf{w}^T\mathbf{X}^T\mathbf{y}} \end{aligned}$$

Examining the above, we need to normalize the equation. The Right hand side of the proportionality above has an exponent that contains:

$$w^T(\lambda\mathbf{I} + \sigma^{-2}\mathbf{X}^T\mathbf{X})\mathbf{w} \tag{1}$$

and

$$2\sigma^{-2}\mathbf{w}^T\mathbf{X}^T\mathbf{y} \tag{2}$$

N.B: (1) is quadratic in w while, (2) is linear in $\mathbf{w}$

To simplify our expressions, let's take $A = \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ and $b = (\frac{1}{\sigma^2})\mathbf{X}^T\mathbf{y}$.
The expressions them becomes $\mathbf{w}^T A\mathbf{w}$ and $2b^T\mathbf{w}$. We can now "complete the quadratic form" by applying following

$$\mathbf{w}^T A\mathbf{w} - 2b^T\mathbf{w} = (\mathbf{w}A^{-1}b)^T A(\mathbf{w}M^{-1}b) - b^T M^{-1}b,$$

which is easily verified by expanding the quadratic form on the RHS
Putting it together, we get

$$\begin{aligned} p(\mathbf{w}|\mathcal{D}) &\propto e^{\frac{1}{2}[(\mathbf{w}A^{-1}b)^T A(\mathbf{w}A^{-1}b)-b^T A^{-1}b]} \\ &= e^{-\frac{1}{2}[(\mathbf{w}A^{-1}b)^T A(\mathbf{w}A^{-1}b)} - e^{-\frac{1}{2}[b^T A^{-1}b]} \\ &= e^{-\frac{1}{2}[(\mathbf{w}A^{-1}b)^T A(\mathbf{w}A^{-1}b)} \end{aligned}$$

Now recall that a Gaussian density in $\mathbf{w}$ is given by

$$N(\mathbf{w}; \mu, \Sigma) = |2\pi\Sigma|^{-\frac{1}{2}} exp^{[-\frac{1}{2}(\mathbf{w}-\mu)^T\Sigma^{-1}(\mathbf{w}-\mu)]}$$

so,

$$p(w|\mathcal{D}) \propto \mathcal{N}(\mathbf{w}; A^{-1}b, A^{-1}) \tag{3}$$

Since the L.H.S and R.H.S of (3) are both densities in w and are proportional, they must be actually equal. Hence,

$$p(\mathbf{w}|\mathcal{D}) = \mathcal{N}(\mathbf{w}; A^{-1}b, A^{-1})$$

where,

$$A = \tfrac{1}{\sigma^2}\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I} \text{ and } b = (\tfrac{1}{\sigma^2})\mathbf{X}^T\mathbf{y}$$

Note that the posterior mean is

$$\begin{aligned} A^{-1}b &= (\frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X} + \Sigma^{-1})^{-1}\frac{1}{\sigma^2}\mathbf{X}^T\mathbf{y} \\ &= (\sigma^2\mathbf{X}^T\mathbf{X} + \sigma^2\Sigma^{-1})^{-1}\mathbf{X}^T\mathbf{y}, \end{aligned}$$

which should look familiar from our study of ridge regression. Indeed, if the prior covariance matrix is taken to be $\Sigma = \frac{\sigma^2}{\lambda}\mathbf{I}$, simplifying

$$\begin{aligned} \Sigma &= \frac{\sigma^2}{\lambda}\mathbf{I} \\ &= (\frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1} \\ &= (\sigma^{-2}\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1})^{-1} \end{aligned}$$

then the posterior mean is

$$= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y},$$

which is exactly the ridge regression estimate for w.
To make things look prettier, people often specify the gaussian prior in terms of the **precision matrix**, which is the inverse of the covariance matrix. That is $\wedge = \Sigma^{-1}$. Then the posterior mean looks like

$$(\mathbf{X}^T\mathbf{X} + \sigma^2\wedge)^{-1}\mathbf{X}^T\mathbf{y}.$$

(Not finished, remains Variance-covariance)

5) Show the maximum posterior estimation of $\mathbf{w}$ is in the ridge regression form $(\mathbf{X}^\top\mathbf{X} + \lambda'\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$.

Soln:
Let's assume that our input data x is already centered so that we can neglect the intercept term $\mathbf{w}_0$. Taking the likelihood as: $y \sim \mathrm{N}(\mathbf{X}\mathbf{w}, \sigma^2 I)$ and the prior: $w \sim \mathrm{N}(0, \tau^2 I)$, the posterior distribution is:

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \frac{1}{Z}p(\mathbf{y}|\mathbf{w}, \mathbf{X})p(\mathbf{w})$$

which can be rewritten as,

$$p(\mathbf{w}|\mathcal{D}) = \frac{1}{Z}p(\mathcal{D}|\mathbf{w})p(\mathbf{w})$$

where $Z = Z(\mathbf{y}, \mathbf{X}) = \int p(\mathbf{y}|\mathbf{w}, \mathbf{X})p(\mathbf{w})d\mathbf{w}$ is the normalization constant that does not depend on $\mathbf{w}$.

To obtain the (log) posterior distribution from the relation clearly:

$$\begin{aligned} p(\mathbf{w}|\mathbf{y}, \mathbf{X}) &= \frac{1}{Z}p(\mathbf{y}|\mathbf{w}, \mathbf{X})p(\mathbf{w}) \\ &= \frac{1}{Z}\frac{1}{(2\pi)^{p/2}\sigma^p}[e^{-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w})}]\frac{1}{(2\pi)^{p/2}\tau}[e^{-\frac{\mathbf{w}^T\mathbf{w}}{2\tau^2}}] \end{aligned}$$

Simplifying,

$$\log p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = -\log Z - K - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) - \frac{\mathbf{w}^T\mathbf{w}}{2\tau^2}$$

where K corresponds to the other terms outside the exponent. Taking the derivative of the $\log p(\mathbf{w}|\mathbf{y}, \mathbf{X})$ with respect to $\mathbf{w}$, we have:

$$\frac{d\log p(\mathbf{w}|\mathbf{y}, \mathbf{X})}{d\mathbf{w}} = \frac{1}{\sigma^2}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) - \frac{\mathbf{w}}{\tau^2}$$

To obtain the maximum, we set the above equation to zero. Hence we have:

$$\begin{aligned} 0 &= \frac{1}{\sigma^2}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) - \frac{\mathbf{w}}{\tau^2} \\ &= \mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) - \frac{\sigma^2}{\tau^2}\mathbf{w} \\ &= \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\mathbf{w} - \frac{\sigma^2}{\tau^2}\mathbf{w} \\ &= \mathbf{X}^T\mathbf{y} - \mathbf{w}(\mathbf{X}^T\mathbf{X} - \frac{\sigma^2}{\tau^2}) \end{aligned}$$

Hence,

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} - \frac{\sigma^2}{\tau^2}\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

Setting the ridge regression parameter $\lambda = \frac{\sigma^2}{\tau^2}$, we can see that the above solution is equivalent to **ridge regression**.
It is also worthy to note how setting the ridge regression parameter λ, varies with τ^2 and σ^2. When we have prior knowledge to believe w is close to zero i.e τ^2 is small, then λ is large. This means that having a large w will be penalized. On the other hand, when σ^2 is small (meaning that observations $\mathbf{y}$ are not noisy), we will focus on fitting the data (small λ).

Question 2 (Implementation, 40 points)

1) Download *prostate* data from
`https://web.stanford.edu/~hastie/ElemStatLearn/data.html`
Take $x_1 = \text{lcavol}, x_2 = \text{lweight}, x_3 = \text{age}, x_4 = \text{lbph}, x_5 = \text{svi}, x_6 = \text{lcp}, x_7 = \text{gleason}, x_8 = \text{pgg45}$, and define the response variable $y = \text{lpsa}$. Standardize all variables. Show the 8 scatter plot of y against each $x_1, \ldots, x_8$.

Soln:

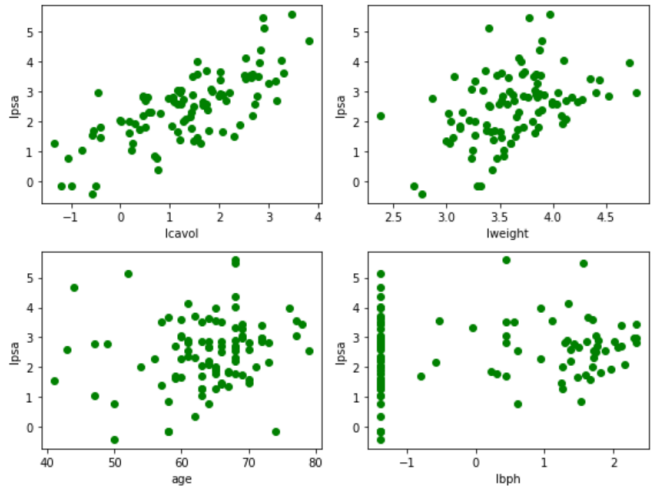
i. Standardize all variables. Values are obtained in the following figure.

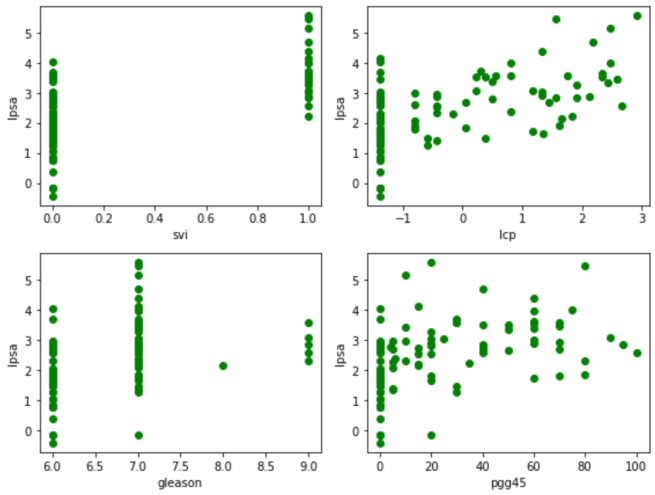
```
[12]  df_standardize = (df - df.mean(axis=0))/df.std(axis=0) #standard
      ✓ 0.6s                                                          Python

[13]  df_standardize.head()
      ✓ 0.7s                                                          Python
```

	lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45	lpsa
0	-1.637356	-2.006212	-1.862426	-1.024706	-0.522941	-0.863171	-1.042157	-0.864467	-2.520226
1	-1.988980	-0.722009	-0.787896	-1.024706	-0.522941	-0.863171	-1.042157	-0.864467	-2.287827
2	-1.578819	-2.188784	1.361163	-1.024706	-0.522941	-0.863171	0.342627	-0.155348	-2.287827
3	-2.166917	-0.807994	-0.787896	-1.024706	-0.522941	-0.863171	-1.042157	-0.864467	-2.287827
4	-0.507874	-0.458834	-0.250631	-1.024706	-0.522941	-0.863171	-1.042157	-0.864467	-1.825150

ii. The scatter plot of y against each $x_1, ..., x_8$ is shown in the following figures.





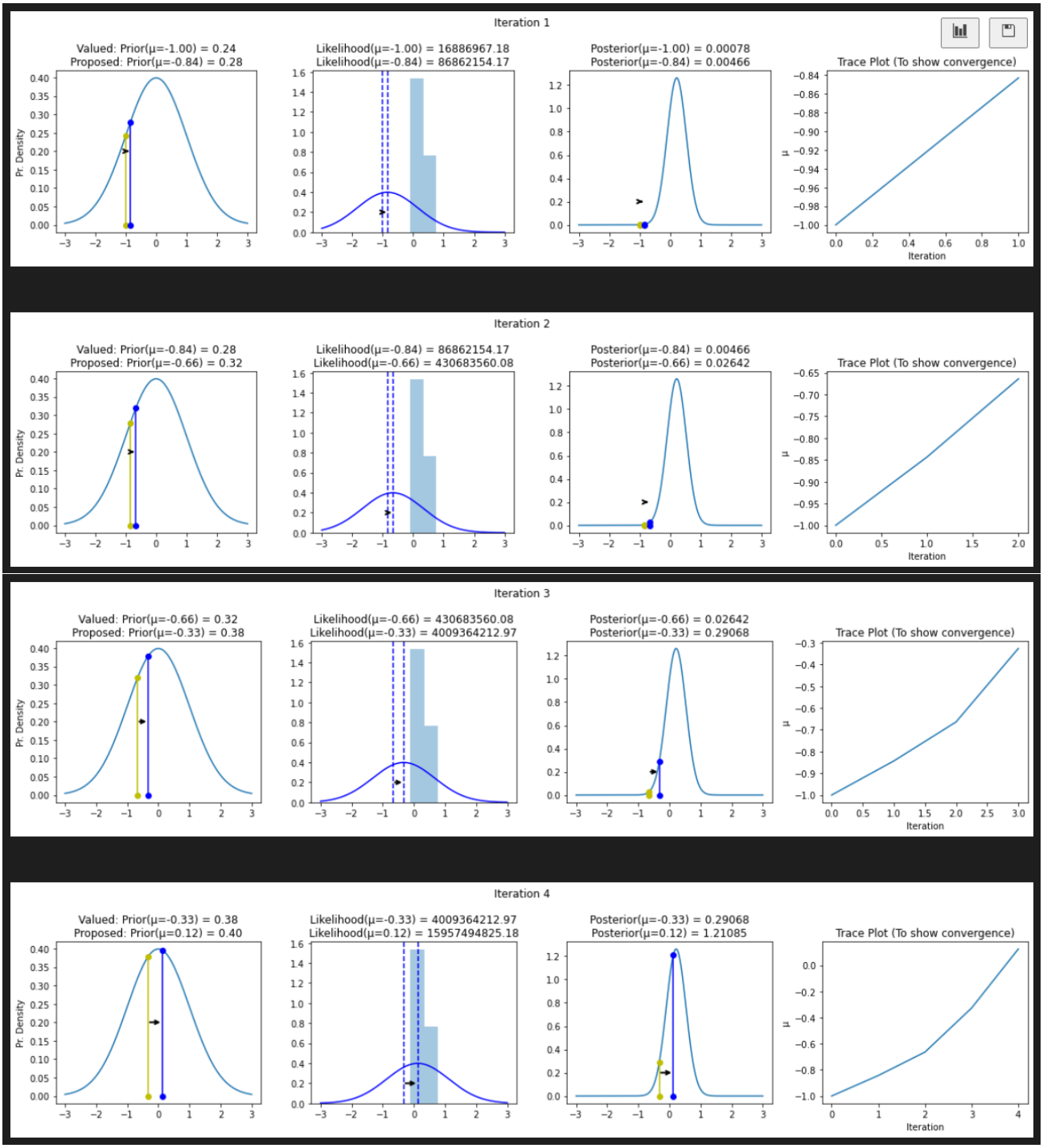
2) Use a Markov chain Monte Carlo sampler to take samples from the posterior of **w** in Question 1 over this data set and visualize your samples, do they converge?

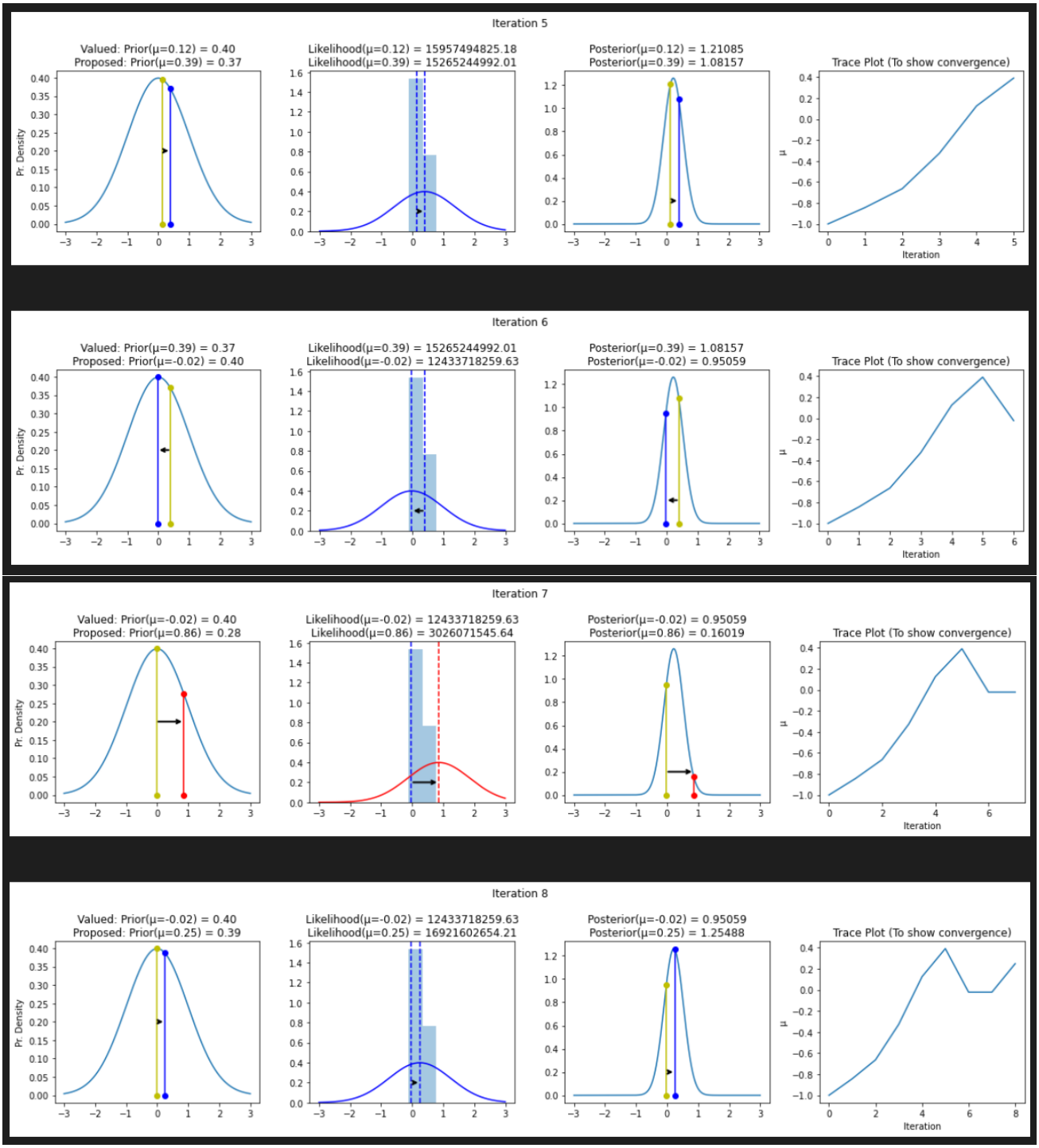
Soln: First we, extract samples from the posterior w in Question 1.

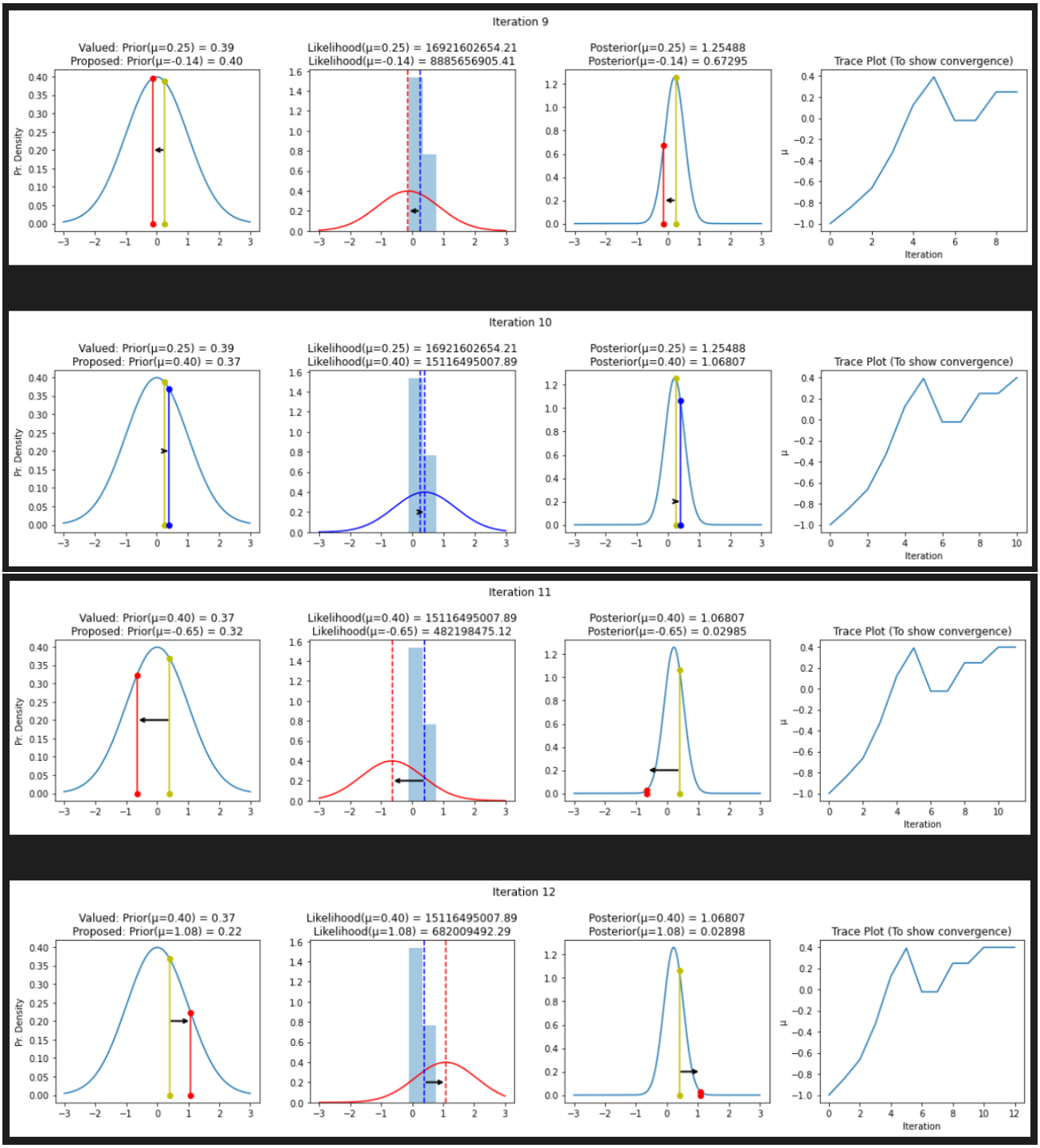
```python
def calc_posterior_analytical(data, x, mu_0, sigma_0):
    sigma = 1.
    n = len(data)
    mu_post = (mu_0 / sigma_0**2 + data.sum() / sigma**2) / (1. / sigma_0**2 + n / sigma**2)
    sigma_post = (1. / sigma_0**2 + n / sigma**2)**-1
    return norm(mu_post, np.sqrt(sigma_post)).pdf(x)
```

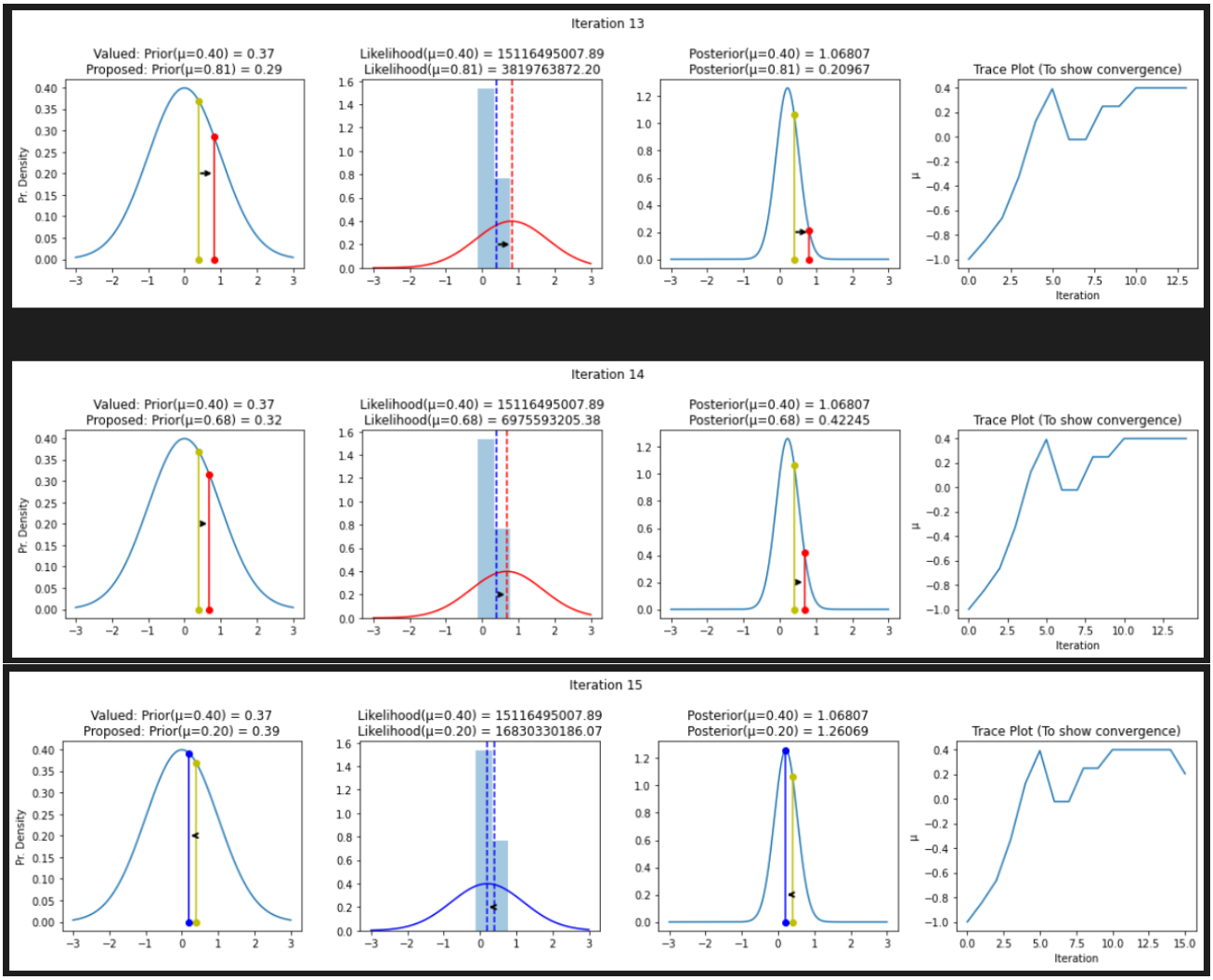
✓ 0.3s Python

The MCMC Sampler used is the Trace plot. The trace plot is a time series plot that shows the realizations of the Markov chain at each iteration against the iteration numbers.


Iteration 1
Valued: Prior(μ=-1.00) = 0.24
Proposed: Prior(μ=-0.84) = 0.28
Likelihood(μ=-1.00) = 16886967.18
Likelihood(μ=-0.84) = 86862154.17
Posterior(μ=-1.00) = 0.00078
Posterior(μ=-0.84) = 0.00466
Trace Plot (To show convergence)
Pr. Density
μ
Iteration
Iteration 2
Valued: Prior(μ=-0.84) = 0.28
Proposed: Prior(μ=-0.66) = 0.32
Likelihood(μ=-0.84) = 86862154.17
Likelihood(μ=-0.66) = 430683560.08
Posterior(μ=-0.84) = 0.00466
Posterior(μ=-0.66) = 0.02642
Trace Plot (To show convergence)
Pr. Density
μ
Iteration
Iteration 3
Valued: Prior(μ=-0.66) = 0.32
Proposed: Prior(μ=-0.33) = 0.38
Likelihood(μ=-0.66) = 430683560.08
Likelihood(μ=-0.33) = 4009364212.97
Posterior(μ=-0.66) = 0.02642
Posterior(μ=-0.33) = 0.29068
Trace Plot (To show convergence)
Pr. Density
μ
Iteration
Iteration 4
Valued: Prior(μ=-0.33) = 0.38
Proposed: Prior(μ=0.12) = 0.40
Likelihood(μ=-0.33) = 4009364212.97
Likelihood(μ=0.12) = 15957494825.18
Posterior(μ=-0.33) = 0.29068
Posterior(μ=0.12) = 1.21085
Trace Plot (To show convergence)
Pr. Density
μ
Iteration


Iteration 5
Valued: Prior(μ=0.12) = 0.40
Proposed: Prior(μ=0.39) = 0.37
Likelihood(μ=0.12) = 15957494825.18
Likelihood(μ=0.39) = 15265244992.01
Posterior(μ=0.12) = 1.21085
Posterior(μ=0.39) = 1.08157
Trace Plot (To show convergence)
Pr. Density
μ
Iteration
Iteration 6
Valued: Prior(μ=0.39) = 0.37
Proposed: Prior(μ=-0.02) = 0.40
Likelihood(μ=0.39) = 15265244992.01
Likelihood(μ=-0.02) = 12433718259.63
Posterior(μ=0.39) = 1.08157
Posterior(μ=-0.02) = 0.95059
Trace Plot (To show convergence)
Pr. Density
μ
Iteration
Iteration 7
Valued: Prior(μ=-0.02) = 0.40
Proposed: Prior(μ=0.86) = 0.28
Likelihood(μ=-0.02) = 12433718259.63
Likelihood(μ=0.86) = 3026071545.64
Posterior(μ=-0.02) = 0.95059
Posterior(μ=0.86) = 0.16019
Trace Plot (To show convergence)
Pr. Density
μ
Iteration
Iteration 8
Valued: Prior(μ=-0.02) = 0.40
Proposed: Prior(μ=0.25) = 0.39
Likelihood(μ=-0.02) = 12433718259.63
Likelihood(μ=0.25) = 16921602654.21
Posterior(μ=-0.02) = 0.95059
Posterior(μ=0.25) = 1.25488
Trace Plot (To show convergence)
Pr. Density
μ
Iteration


Iteration 9
Valued: Prior(μ=0.25) = 0.39
Proposed: Prior(μ=-0.14) = 0.40
Likelihood(μ=0.25) = 16921602654.21
Likelihood(μ=-0.14) = 8885656905.41
Posterior(μ=0.25) = 1.25488
Posterior(μ=-0.14) = 0.67295
Trace Plot (To show convergence)
Pr. Density
μ
Iteration
Iteration 10
Valued: Prior(μ=0.25) = 0.39
Proposed: Prior(μ=0.40) = 0.37
Likelihood(μ=0.25) = 16921602654.21
Likelihood(μ=0.40) = 15116495007.89
Posterior(μ=0.25) = 1.25488
Posterior(μ=0.40) = 1.06807
Trace Plot (To show convergence)
Pr. Density
μ
Iteration
Iteration 11
Valued: Prior(μ=0.40) = 0.37
Proposed: Prior(μ=-0.65) = 0.32
Likelihood(μ=0.40) = 15116495007.89
Likelihood(μ=-0.65) = 482198475.12
Posterior(μ=0.40) = 1.06807
Posterior(μ=-0.65) = 0.02985
Trace Plot (To show convergence)
Pr. Density
μ
Iteration
Iteration 12
Valued: Prior(μ=0.40) = 0.37
Proposed: Prior(μ=1.08) = 0.22
Likelihood(μ=0.40) = 15116495007.89
Likelihood(μ=1.08) = 682009492.29
Posterior(μ=0.40) = 1.06807
Posterior(μ=1.08) = 0.02898
Trace Plot (To show convergence)
Pr. Density
μ
Iteration



Observation: This graphical method is used to visualize how the Markov chain is moving around the state space, that is, how well it is mixing. If the MCMC chain is stuck in some part of the state space, the trace plots shows flat bits indicating slow convergence. From 9^{th} iteration, we noticed a convergence in the result of the trace plots. Thus, We can say that they converge due to this.

3) Approximate the maximum posterior using your MCMC samples and compare your MCMC approximation with the theoretical maximum of Question 1.5

Soln: Recall that, we would obtain the posterior **w** (theoretical) from question 1.5 as shown below.

```
w
✓ 0.3s                                                    Python
array([ 0.18156085,  0.56434128,  0.62201979, -0.02124819,  0.09671252,
        0.7616734 , -0.10605094,  0.04922793,  0.00445751])
```

Using K-Fold cross-validation (and also the inbuilt ridge method) to determine the appropriate learning rate, λ that fits the model

```
def ridge(learning_rates):
    # Create an empty data frame
    data = pd.DataFrame()
    data['Feature Name'] = df.columns.drop("lpsa")

    # For each alpha value in the list of alpha values,
    for learning_rate in learning_rates:
        ridge = Ridge(alpha=learning_rate, random_state=42)
        ridge.fit(df.drop("lpsa", axis = 1), df["lpsa"])
        # Create a column name for that alpha value
        column_name = 'lr = %f' % learning_rate
        # Create a column of coefficient values
        data[column_name] = ridge.coef_

    # Return the dataframe
    return data

ridge(learning_rates)
0.6s                                                      Python
```

	Feature Name	lr = 0.000000	lr = 0.001000	lr = 0.010000	lr = 0.100000	lr = 1.000000	lr = 10.000000	lr = 100.000000	lr = 1000.000000	lr = 2000.000000	lr = 5000.000000
0	lcavol	0.564341	0.564341	0.564342	0.564345	0.563762	0.532865	0.301768	0.062290	0.033454	0.014157
1	lweight	0.622020	0.621979	0.621608	0.617932	0.583576	0.382318	0.106300	0.016211	0.008500	0.003540
2	age	-0.021248	-0.021247	-0.021239	-0.021155	-0.020372	-0.015395	-0.003120	0.005588	0.005732	0.004703
3	lbph	0.096713	0.096714	0.096727	0.096858	0.098121	0.104986	0.079876	0.018223	0.009925	0.004294
4	svi	0.761673	0.761588	0.760821	0.753239	0.685508	0.373626	0.091962	0.015356	0.008156	0.003440
5	lcp	-0.106051	-0.106031	-0.105851	-0.104073	-0.087804	0.001383	0.106710	0.036151	0.020200	0.008876
6	gleason	0.049228	0.049217	0.049115	0.048109	0.039376	0.009284	0.009695	0.003444	0.002048	0.001052
7	pgg45	0.004458	0.004458	0.004459	0.004473	0.004591	0.004970	0.007392	0.014162	0.015085	0.015337

We would observe (from column $lr = 0.1$ to $lr = 1$), we can select the value of the λ that can be compared the the posterior $\mathbf{w}$ obtained earlier. Any learning rate with $\lambda > 1$ would overfit for the array of posterior $\mathbf{w}$ obtained earlier.

4) Compute principal components of $x_1, \ldots, x_8$, plot the data on first the two principal axes z_1, z_2, and interpret your finding.

Soln: 1. We use the method of obtaining the ‘eigenvalues‘, ‘eigenvectors‘ and ‘covariance‘ to solve the principal components.

```python
cov = np.cov(df_centered.iloc[:, :-1], rowvar = False)
evals , evecs = np.linalg.eig(cov)

# sort results wrt. eigenvalues
idx = evals.argsort()[::-1]
evals, evecs = evals[idx], evecs[:, idx]

# projections of X on the principal axes are called principal components
principal_components = df_centered.iloc[:, :-1].dot(evecs)
principal_components.columns = ["z1", "z2", "z3", "z4", "z5", "z6", "z7", "z8"]
from sklearn.decomposition import PCA
pca = PCA(n_components = 8)
X2D = pca.fit_transform(df.iloc[:, :-1])
pd.DataFrame(X2D, columns = ["z1", "z2", "z3", "z4", "z5", "z6", "z7", "z8"])
principal_components
```

✓ 0.8s Python

2. We use the Singular value decomposition technique (SVD) shown in the following figure.

Using **Singular Value Decomposition (SVD)** technique to decompose the training set matrix `df_standardize` into the matrix multiplication of three matrices $U\ \Sigma,\ V^T$, where `V` contains the unit vectors that define all the principal components that we are looking for.

```python
df_centered = df - df.mean() #mean centering
U, sigma, Vt = np.linalg.svd(df_centered.iloc[:, :-1]) #response variable not included
W2 = Vt.T[:, :]
```

✓ 0.6s Python

```python
#print out the principal components
pc = df_centered.iloc[:, :-1].dot(W2)
pc.columns = ["z1", "z2", "z3", "z4", "z5", "z6", "z7", "z8"]
pc
```

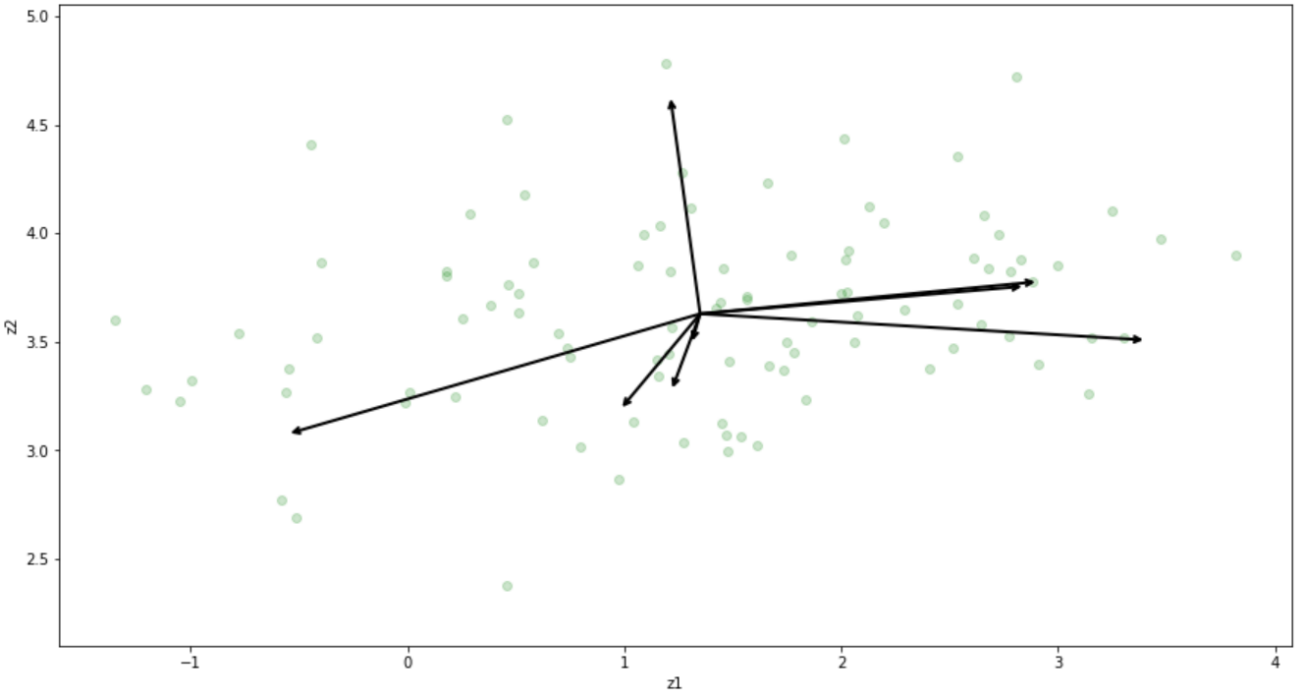
Python

The final results obtained from the methods is given below.

	z1	z2	z3	z4	z5	z6	z7	z8
0	-25.459202	12.030635	-0.538722	1.335912	-0.574250	-0.012538	-0.368914	0.122934
1	-24.845056	4.074839	-0.383910	1.917513	-1.002525	0.089402	0.122153	0.049582
2	-3.654424	-10.273257	0.109555	3.146353	-0.743257	-0.604385	-0.513588	0.156981
3	-24.848936	4.079094	-0.484650	2.022187	-1.154771	0.061686	0.114518	0.063587
4	-24.502729	0.067452	0.645971	1.303411	0.170393	0.276855	-0.039887	-0.044356
...	...	...	...	...	...	...	...	...
92	35.878536	-1.243826	1.853127	0.595723	0.235356	0.627420	0.179679	0.275541
93	14.138598	21.048719	2.189831	-2.165541	0.704699	0.286596	0.361884	0.176673
94	-15.132307	10.792222	3.091581	-2.077010	-0.585431	-0.461103	0.001748	0.305018
95	55.821790	0.105133	-0.581274	-1.122968	0.226914	0.871527	-0.312716	0.334446
96	-3.896732	-4.479730	2.780688	-2.765426	-0.720277	-0.157170	0.068955	0.162783

97 rows × 8 columns

Plotting the data of the first two principal axes z_1, z_2



Interpretation of Findings: From the plot, 2 Principal Components (PCs) seems to carry

much information and it's save to reduce the dimensionality of the training predictor into 2 PCs, and this can be verified below.

```
pca.explained_variance_ratio_[:8]
```
✓ 0.5s Python

```
array([9.34850309e-01, 5.97407213e-02, 2.29917376e-03, 2.05144759e-03,
       5.57711461e-04, 2.61499680e-04, 1.41613378e-04, 9.75236216e-05])
```

This figure above shows the proportion of the dataset's variance that lies along each principal component. From the figure, we obtained: $PC_1 = 9.34850309e-01$, $PC_2 = 5.97407213e-02$, and other PCs as 2.29917376e-03, 2.05144759e-03, 5.57711461e-04, 2.61499680e-04, 1.41613378e-04, 9.75236216e-05. This output tells you that 93.5% (i.e 0.93485031) of the dataset's variance lies along the first PC, and just 5.97% (i.e 0.05974072) lies along the second PC. This leaves less than 0.541% for the other Principal Components.
Therefore, it is reasonable to assume that the 3^{rd}, 4^{th}, 5^{th}, 6^{th}, 7^{th} and 8^{th} PCs probably carries little/no information.