1.

I.

Let h_i denotes for the hashed vector of x_i

$$h_i = Ax_i = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{p1} & \dots & a_{pn} \end{bmatrix} \begin{bmatrix} x_{i1} \\ \dots \\ x_{in} \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{n} a_{1j}x_{ij} \\ \dots \\ \sum_{j=1}^{n} a_{kj}x_{ij} \\ \dots \\ \sum_{j=1}^{n} a_{pj}x_{ij} \end{bmatrix}$$

Now, we will prove that for all $k \ in \ [1, p]$, $\mathrm{P}\left[\sum_{j=1}^{n} a_{kj}x_{ij} = 1\right] = \mathrm{P}\left[\sum_{j=1}^{n} a_{kj}x_{ij} = 0\right] = \frac{1}{2}$.

First, divide j into two sets S_1 and S_2. For all $j \in S_1$ $x_{ij} = 1$, for all $j \in S_2$ $x_{ij} = 0$.

Thus we could rewrite the value as

$$\sum_{j=1}^{n} a_{kj}x_{ij} = \sum_{j\in S_1} 1[a_{kj}] + \sum_{j\in S_2} 0[a_{kj}] = \sum_{j\in S_1} a_{kj}$$

Now what we need to prove becomes $\mathrm{P}\left[\sum_{j\in S_1} a_{kj} = 1\right] = \mathrm{P}\left[\sum_{j\in S_1} a_{kj} = 0\right] = \frac{1}{2}$.

Suppose the size of S_1 is n_{s1} $(0 < n_{s1} \le n)$. Rename $a_{kj},\ j \in S_1$ as $d_1, d_2 \dots\ d_{n_{s1}}$, we have

$$\sum_{j\in S_1} a_{kj} = \sum_{t=1}^{n_{s1}} d_t$$

a. If $n_{s1} = 1$, then $\sum_{t=1}^{n_{s1}} d_t = d_1$ as d_1 is uniformly randomly picked form {0, 1}, $\mathrm{P}[d = 1] = \mathrm{P}[d = 0] = \frac{1}{2}$ and $\mathrm{P}\left[\sum_{j=1}^{n} a_{kj}x_{ij} = 1\right] = \mathrm{P}\left[\sum_{j=1}^{n} a_{kj}x_{ij} = 0\right] = \frac{1}{2}$.

b. If $n_{s1} > 1$, $\sum_{t=1}^{n_{s1}} d_t = d_{n_{s1}} + \sum_{t=1}^{n_{s1}-1} d_t$. As d_t is independent,

$$\mathrm{P}\left[\sum_{t=1}^{n_{s1}} d_t = 1\right] = \mathrm{P}\left[d_{n_{s1}} = 1\right]\mathrm{P}\left[\sum_{t=1}^{n_{s1}-1} d_t = 0\right] + \mathrm{P}\left[d_{n_{s1}} = 0\right]\mathrm{P}\left[\sum_{t=1}^{n_{s1}-1} d_t = 1\right]$$

$$\mathrm{P}\left[\sum_{t=1}^{n_{s1}} d_t = 0\right] = \mathrm{P}\left[d_{n_{s1}} = 0\right]\mathrm{P}\left[\sum_{t=1}^{n_{s1}-1} d_t = 0\right] + \mathrm{P}\left[d_{n_{s1}} = 1\right]\mathrm{P}\left[\sum_{t=1}^{n_{s1}-1} d_t = 1\right]$$

Same as above, $d_{n_{s1}}$ is uniformly randomly picked form {0, 1}, so $\mathrm{P}\left[d_{n_{s1}} = 1\right] = \mathrm{P}\left[d_{n_{s1}} = 0\right] = \frac{1}{2}$, and thus

$$\mathrm{P}\left[\sum_{t=1}^{n_{s1}} d_t = 1\right] = \frac{1}{2}\left\{\mathrm{P}\left[\sum_{t=1}^{n_{s1}-1} d_t = 0\right] + \mathrm{P}\left[\sum_{t=1}^{n_{s1}-1} d_t = 1\right]\right\} = \frac{1}{2}$$

$$\mathrm{P}\left[\sum_{t=1}^{n_{s1}} d_t = 0\right] = \frac{1}{2}\left\{\mathrm{P}\left[\sum_{t=1}^{n_{s1}-1} d_t = 0\right] + \mathrm{P}\left[\sum_{t=1}^{n_{s1}-1} d_t = 1\right]\right\} = \frac{1}{2}$$

So we have proven for all $k\ in\ [1,p]$, $\mathrm{P}[\sum_{j=1}^{n} a_{kj}x_{ij} = 1] = \mathrm{P}[\sum_{j=1}^{n} a_{kj}x_{ij} = 0] = \frac{1}{2}$.

Once again because entries of A are independent, thus

$$\mathrm{P}[h_i = b] = \prod_{k=1}^{p} \mathrm{P}[\sum_{j=1}^{n} a_{kj}x_{ij} = b_k] = \frac{1}{2^p}$$

II.

There are total number of 2^p kinds of different hash vectors b, thus

$$\mathrm{P}\left[h_i = h_j\right] = \sum_{k=1}^{2^p} \mathrm{P}[h_i = b_k]P[h_j = b_k] = \frac{2^p}{2^{2p}} = \frac{1}{2^p}$$

III.

Let's examine the probability of collision happens among x_i.

As proven in II, for any $1 \le i < j \le m$, probability of collision happens between x_i and x_j is $\frac{1}{2^p}$.

Thus,

$$\mathrm{P}[collision\ happens] = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \mathrm{P}\left[h_i = h_j\right] = \sum_{t=1}^{m-1} \frac{1}{2^p} = \frac{m(m-1)}{2}\frac{1}{2^p}$$

Now that $p \ge 2log_2 m$,

$$\mathrm{P}[collision\ happens] = \frac{m(m-1)}{2}\frac{1}{2^p} \le \frac{m(m-1)}{2}\frac{1}{m^2}$$

$$\mathrm{P}[collision\ happens] \le \frac{1}{2}\frac{m-1}{m}$$

$$\mathrm{P}[collision\ happens] < \frac{1}{2}$$

Thus, if we set $p \ge 2log_2 m$, we have at most $\frac{1}{2}$ chance of collision happens, which means there is at least $\frac{1}{2}$ chance of no collision happens. The question is proven.

2.

Let y be the dependent variable and w be the weight vector

Joint probability distribution of y, w and x is given by $P(w, y, x)$

With chain rule of joint probability, we have

$$P(w, y, x) = P(w|y, x)P(y, x) = P(y|w, x)P(w, x)$$

$$P(w|y, x) = \frac{P(y|w, x)P(w, x)}{P(y, x)}$$

$$\propto\ P(y|w, x)P(w, x)$$

$$\propto P(y|w,x)P(w) \qquad \text{equation ①}$$

Since the probability distribution is Gaussian,

$$P(y|w,x) = N(y - wx, 0, \tau)$$

Therefore, we could write y_t as follows

$$y_t = wx_t + \varepsilon \;\; where\; \varepsilon = N(0, \sigma)$$

The optimization problem for the equation ① could be written as maximize the following function

$$L(w) = \prod_{t=1}^{T} N(y_t - wx_t, 0, \tau)N(w, 0, \sigma)$$

To convert the above maximization problem into a minimization problem, take the logs on both sides of the above equation

$$\ln(L(w)) = \sum_{t=1}^{T} \ln[N(y_t - wx_t, 0, \tau)] - \ln[N(w, 0, \sigma)]$$

We know that N(x, μ, σ) = $\frac{1}{\sqrt{2\pi\sigma^2}}$ $\exp(-(x-\mu)^2/2\sigma^2)$

$$\ln(L(w)) = \sum_{t=1}^{T} \frac{(y_t - wx_t)^2}{2\tau^2} + \frac{w^2}{2\sigma^2} \qquad \text{equation ②}$$

Moving to ridge regression, the objective of ridge regression is to minimize

$$||Xw - y||^2 + \lambda||w||^2 = \sum_{t=1}^{T}(y_t - wx_t)^2 + \lambda w^2 \qquad \text{equation ③}$$

If we compare the equations ① and ③, we see that it is equivalent. It proves that finding the coefficients w is equivalent to minimizing the ridge regression

3.

For this part, we made many attempts and had several algorithms that works well.

a. **Attempt of Xiaoning Wan: (Team Name on kaggle: Xiaoning_Wan)**

Algorithm:

The algorithm I used is neural networks. The basic idea is simple, to extract features from raw data, and train the model according to these features.

After some researches, I decided to use bag-of-words as the feature.

i. That is to create a dictionary which contains all the words and digits (for digits, use the length of each digit as the 'word', for example the 'word' corresponding to 1000 is 4 and the 'word' corresponding to 522 is 3) appeared in ham e-mails and spam e-mails.
ii. Then, calculate the frequency of words in each e-mail. Use this frequency as the feature of e-mails.

Now with the feature, I could build a neural network for classification.

Model:

The neural network I built has three layers, two dense layer and one output layer.

The first dense layer has 150 neurons, the second dense layer has 50 neurons, their activation functions are both 'ReLu'

(for details about 'ReLu': https://en.wikipedia.org/wiki/Rectifier_(neural_networks))

The output layer uses soft-max for classification.

The optimizer used is Adam optimizer, learning rete is set to 0.001, batch size is 100, number of epochs is 100.

Performance:

The highest validation accuracy over 100 epochs is 97.254%

The accuracy of given test e-mails is 97.499%.

b. Attempt of Smitha and Sravya: (Team Name on kaggle: xw2501_se2444_spy2107)

The word embedding technique we used was bag of words. We initially did this for this first for the simplicity of the implementation. It gave us a high accuracy (97%), so we stuck with this representation. In the future, we could try use something that preserves the context of the words better, such as word2vec. The classifier that we used was a random forest. After researching this sort of problem more and speaking with the professor, we found that ensemble methods, in which there are multiple classifiers that "vote" on the final classification, produce the best accuracy. Thus, we chose to try the Random Forest ensemble method.

In order to pre-process the data, we first read the data from each file into an array. We then separated the words in each email by spaces. After that, we applied the following preprocessing techniques to obtain the meaningful words from each email to be used as features in our bag of words:

1. Converting everything to lowercase: We did this to ensure that the same word wouldn't be counted differently in the bag of words representation based on lowercase/uppercase
2. Removing non-letters: We tried both with keeping numbers and punctuation and removing them. We ended up with the same test accuracy on both, so we opted to remove non-letters from our feature list.
3. Removing stop words: Stop-words, such as "the" don't give much information, so we didn't want them to be used as features in the bag of words representation.
4. Stemming words: We did this to ensure that the same word in a different tense (e.g. read and reading) would not be categorized as different words. We used a lemmatizer from NLTK to ensure that this was done correctly.

The resources we used were:

- NLTK - WordNetLemmatizer, stopwords, word stemmer, word tokenizer
- SKlearn - RandomForestClassifier, CountVectorizer
- Pandas
- BeautifulSoup

Our initial implementation using bag of words and a random forest classifier gave us a high accuracy, so we did not need to make many improvements. However, we did play around with the preprocessing techniques mentioned above.