CS5785 / ORIE5750 / ECE5414 - Hw1

Jiwon Jeong

September 2025

1 Warm Up

Code Location: warm_up.py, warm_up.ipynb

This section was mainly using different library operations from numpy and PyTorch. Most questions were direct uses of the library API.

However, for question 3, I learned a lot about how to use tensor and autograd to compute the gradient by following the examples in lecture. Especially for 3a and 3c, it was interesting to see how multivariable equations can be represented as vector operations, including the dot product, on the input dimensions.

1.1 Q1

I successfully checked my reshaping with the .shape property of numpy arrays.

1.2 Q2

I computed each operation using one-line operations of two tensors.

I got tensor([6, 9, 13, 15]), tensor([5, 18, 40, 54]), tensor([1, 729, 390625, 10077696]), tensor(117), and tensor(1235.4036).

1.3 Q3a

I represented the complex function using tensors by thinking of the tensors as vectors. I started with the input vector (tensor) times the input vector times the elementwise exponentiation of the input vector. Then I took the dot product of this vector and the vector [1,3,5,6] to represent the coefficients and summation in the equation. I got the following gradient tensor([5194.4609, 58093.7500, 1192383.2500, 4813232.0000]).

1.4 Q3b

I represented the equation as pairs of different two matrix multiplications (like $A * B^T$ and $A * A^T$). Then I kept multiplying the resulting matrices together. Lastly, I took the L2 norm of the multiplied matrices. Squared this value and took the logarithm of it to get tensor([[0.1737, 0.2854], [0.3837, 0.6404]]).

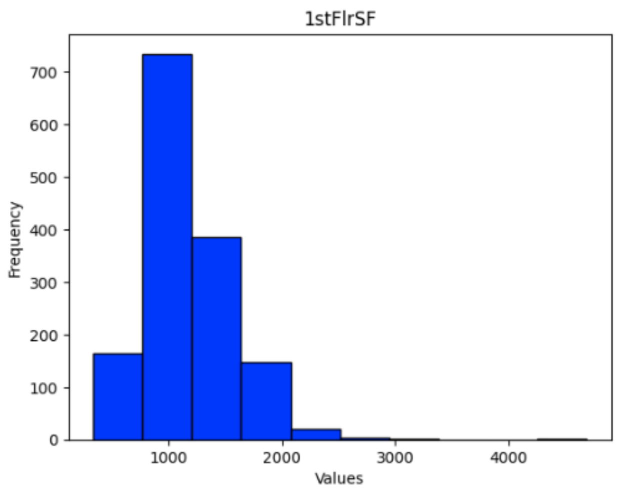


Figure 1: Distribution of First Floor Area Attribute

1.5 Q3c

I represented the tanh(x) function using its exponential definition. With this definition, I was able to reconstruct the function using the same tools described in Q3 and Q4 (torch operations on a torch tensor). I got the gradient of tensor([9.8661e-03, 3.2187e-06]) .

1.6 Q4-Q5

I used a simple PyTorch tensor method tensor.numpy() to conver the tensor into a numpy array.n Similarly for Q5, I directly used numpy methods like numpy.matmul() and numpy.linalg.norm() do perform the operations. For Q5, I got [[27 57] [29 49]] , and 100.024996875781.

2 Housing Prices

2.1 Initial Steps

Code Location: preprocessing/preprocessing.ipynb, preprocessing/graphing.py

I started this competition by reading the data set description and finding 3 categorical and 3 continuous attributes. Some continuous attributes I found were the first floor area (1stFlrSF), the second floor area (2ndFlrSR), and the lot size (LotArea). Some categorical attributes I found were the garage location (GarageType), if central air existed (CentralAir), and the heating quality (HeatingQC). I visualized the distribution of 1stFlrSF (Figure 1) and GarageType (Figure 2).

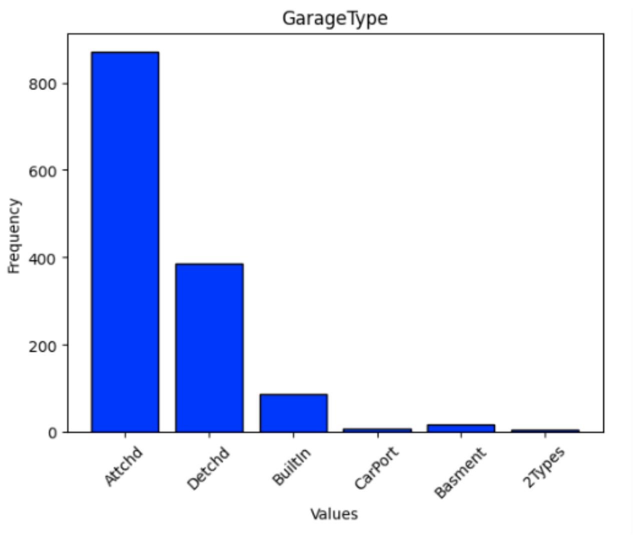


Figure 2: Distribution of Garage Location Attribute

2.2 Pre-processing

Code Location: preprocessing/preprocessing.ipynb

I approached nearly all of my preprocessing steps with the help of Kaggle's feature engineering and data cleaning courses.

2.2.1 Dealing with missing values

I started with addressing missing values, because I learned that models do not know how to handle missing values well. Before pre-processing there was 6.62% of values as "NA", meaning missing.

By using the data description, I determined that many of the attributes like Alley and BsmtQual intentionally had values of "NA" to indicate that the alley/basement is missing. For these attributes, I replaced all the "NA" values with "None", changing nothing semantically but allowing the models to understand that it is an intentional value.

For the three attributes, MasVnrType, MasVnrArea, and GarageYrBlt, I realized that they were intentionally not recorded if the masonry veneer or garage did not exist. For example, instead of a missing masonry veneer having a size of 0, it was recorded as 'NA'. For these, I used imputation to assign 0 or "None" values. For GarageYrBlt specifically, I think that imputating to 0 does not make sense in a linear model, but I did not know how else to handle the missing values.

Lastly, two attributes, LotFrontage and Electrical, had actual missing values. For LotFrontage, I dropped the attribute since it had 259 missing values. For Electrical, it only have 1 missing value, so I instead (temporarily) dropped the single row. The dropped row is used in the final model (which happened to not use the Electrical feature, so the missing value did not matter).

2.2.2 Dealing with categorical values and one-hot encoding

I used two methods to handle categorical values: one-hot encoding and target encoding. I preferred encoding over dropping categorical attributes, because I predicted that some like GarageQual and Neighborhood would be very important to the price based on domain experience.

I preferred to use one-hot encoding (OHE) for most categorical values, especially if the values had no semantic linear relationship. For example, RoofStyle has values of Flat, Gable, Gambrel, Hip, Mansard, and Shed. Value encoding by mapping these categories to 1, 2, 3, 4, 5, 6 does not make sense since these categories do not have any semantic linear relationship. Other good examples of categorical values to OHE are SaleCondition, RoofMat1, Foundation, and Condition2. Additionally, these attributes have low cardinality (not many possible values) and would not cause severe memory issues when doing OHE.

However, even for categorical attributes that could have values with some semantic linear relationship, like those ranking qualities from poor to excellent, I decided to use OHE to not impose incorrect value mapping (and linearity) on the scale. For example for ExterQual, a value encoding of poor to 1 and excellent to 5 could be done. However, I think that exterior quality going from poor to fair is very very important, while exterior quality going from good to excellent is not very important. These domain-specific nuances cannot be captured by value encoding, so I preferred OHE.

I visualized the frequency of values in one categorical attribute, GarageQual, and the number of non-zero values in the one-hot encoded GarageQual features (Figure 3). As expected, the distributions are the same, since OHE creates one non-zero value in the corresponding OHE feature each time that category appears in the categorical attribute.

I did not use OHE for attributes with cardinality greater than 10 to avoid memory issues. For these four attributes, Neighborhood, Exterior1st, Exterior2nd, and MSSubClass, I used value encoding. Value encoding works by assigning the categorical value to the average of the label value (SalePrice) of the categories of the attribute. To reduce the effect of variance on the label means of rare categories, I used smoothing. Smoothing applies a weighted average between the global label mean and the categorical value label mean. It applies a heavier contribution to the global mean if the category is rare or missing in the training set.

2.2.3 Scaling values

Lastly I scaled all my features (which are now numerical) to between 0 and 1. While I don't think this will change the performance of the model, doing this step adds interpretability. The parameters of the model will be comparable to

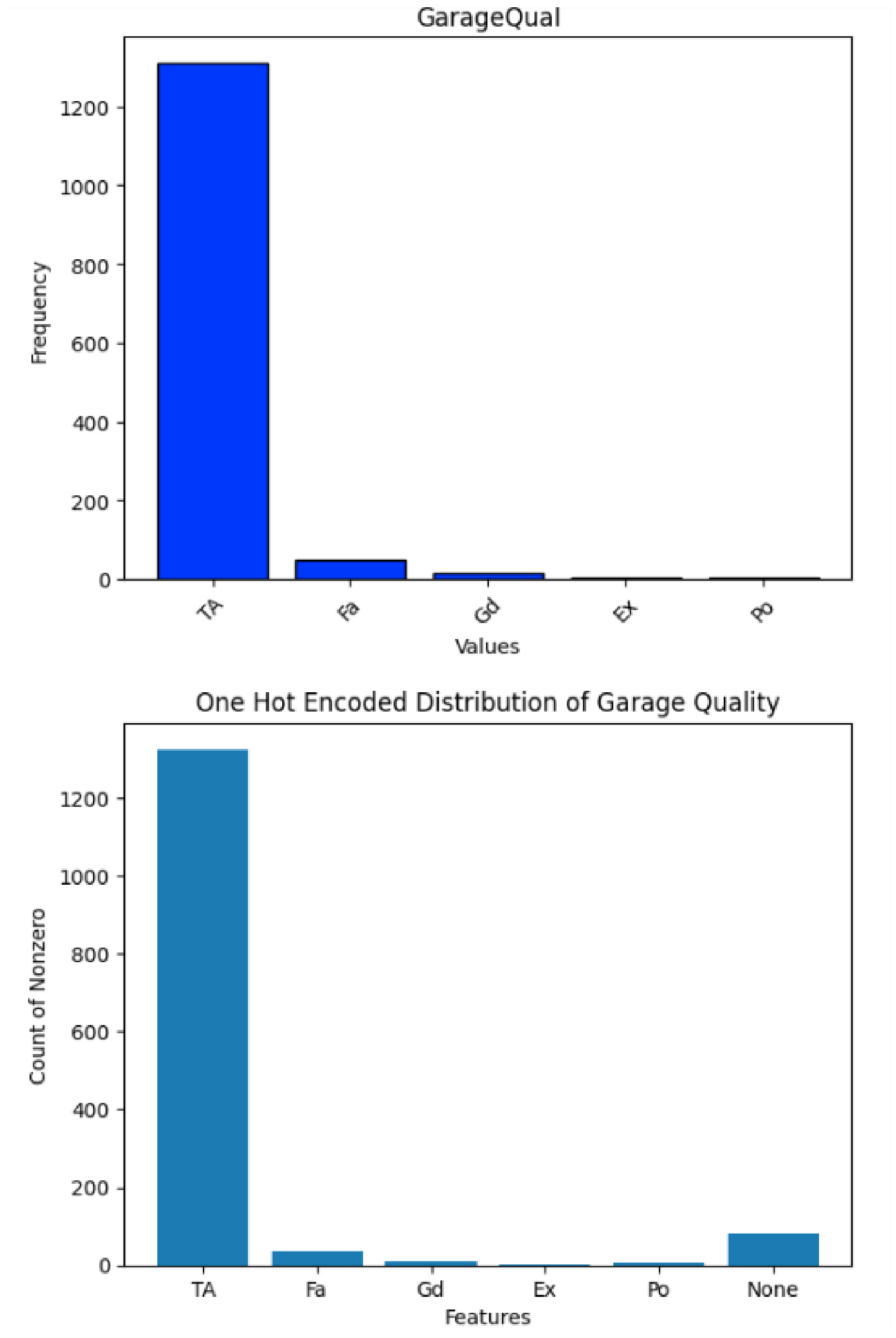


Figure 3: Top: Frequency of each categorical value in GarageQual, Bottom: Number of non-zero in each OHE of GarageQual

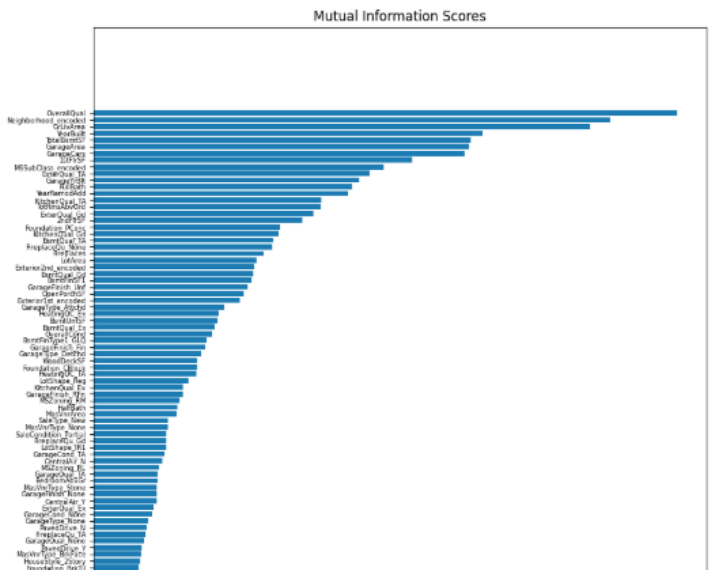


Figure 4: Mutual information of features

each other, since the scaling step means the parameters are relative scalars on the features.

I did not normalize any of my data. The distinction between normalization and scaling is that normalization changes the distribution to fit a Gaussian (or other) distribution better. I decided that a linear regression does not require specifically distributed features.

Note that I did the scaling of values after feature selection and addition discussed in the below section.

2.3 Selecting features

I relied on Kaggle's feature engineering course to guide my feature selection. All of this feature engineering is done using the training data.

2.3.1 Mutual information

I used mutual information (MI) to identify the most relevant features to consider. I knew that I did not want to use all the features, since this would lead to overfitting. The MI of features showed a fast, exponential decrease, meaning that features beyond even the top 30th had low MI values (Figure 4).

The shape of the MI scores curve showed that there is no clear cut-off where MI scores drastically drop. Therefore, I decided to choose the top 10, 20, 30, 40, and 50 MI features and compare models made from each of these. I based these numbers from how I felt $\tilde{2}5$ was where the MI scores were low.

2.3.2 Creating features and domain context

I looked at the top 50 MI feature subset and considered any combination features based on domain experience. I could only identify three. YearBuilt, GarageYrBlt, and YearRemodAdd are all numerical features with values as the year for when the house was built, when the garage was built, and when the house was remodeled. However, buyers don't think about house value as what year it was built, but rather how many years it has been since the house was built. Thus, I wanted to subtract the year sold, YrSold, from each of these features. Since linear models can learn about sum/difference combinations on their own, I simply added back the YrSold feature.

2.3.3 Visualizing relationships

Since MI only is a measure of relationship, not linear relationship, I also wanted to see the linearity of the chosen features. The goal was to identify any high MI features with low linearity. Then to visualize these features and see if there is some other type of relationship (like exponential) that I could apply transformations to make linear now.

The first step was to visualize the linearity of each high MI feature with the label (SalePrice) through a correlation. As seen, there are a few features that have very poor linearity, like YrSold and OverallCond (Figure 5).

I selected all of the low correlation (¡0.3) features and visualized them. Looking at the plots, I realized that the source of low correlation was more likely from high variance and weak relationship with the label, rather than a nonlinear relationship. This aligned with the fact that the MI scores were low for many of the top 50, so I may be visualizing some of these poorly correlated features. An example of a poorly correlated feature shows that there is no clear positive or negative (or even nonlinear) association (Figure 6). Thus, for the plots with visibly poor correlation, I removed them from the feature set. I removed BsmtQual_Gd, GarageFinish_RFn, and LotShape_Reg. There were some features that I saw some potential correlation and decided to keep (Figure 7).

2.3.4 Handling the testing data

Now that there are decided sets of features and clear pre-processing rules, I applied the steps onto the training data. First, I determined what attributes are needed for the largest feature set and filtered for all those attributes from the training data. Then, I put the training data through a small pipeline that automates the pre-processing that I did manually. When creating the pipeline, I tested the pipeline on the training data and validated the results to my manual pre-processing. Lastly, I took the pre-processed features and filtered for the exact feature sets that I decided on through feature engineering.

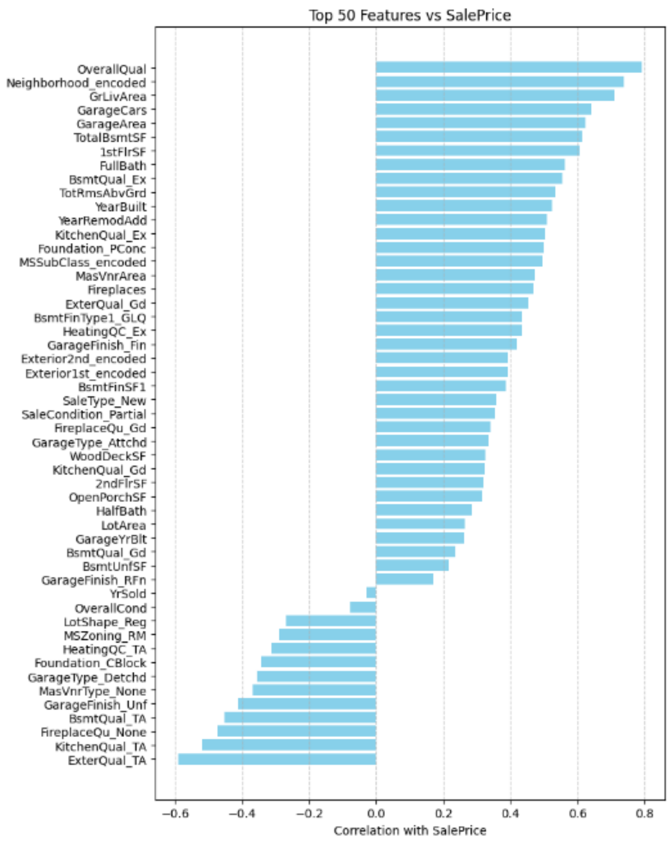


Figure 5: Correlation with SalePrice of high mutual information features

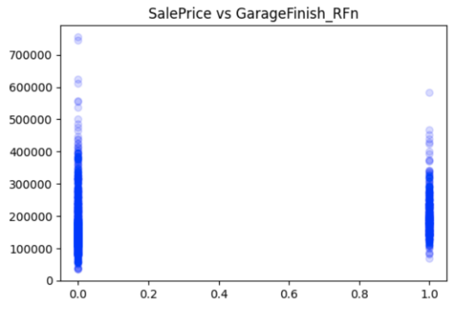


Figure 6: Scatterplot of a feature that I decided to remove

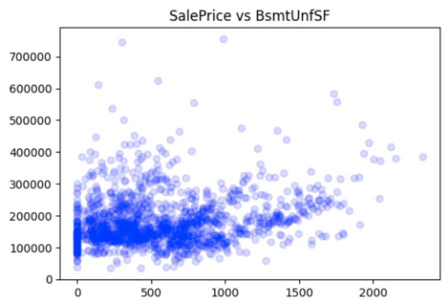


Figure 7: Scatterplot of a feature that I decided to keep

2.4 Training and Testing Linear Regression

2.4.1 Feature selection

As discussed, feature selection was decided by a combination of MI values, domain-specific context, and visualizing relationships. Due to the MI distribution, I decided on 5 tiers of feature sets with increasing number of (potentially less relevant) features. To choose between these sets, I trained and evaluated models on each.

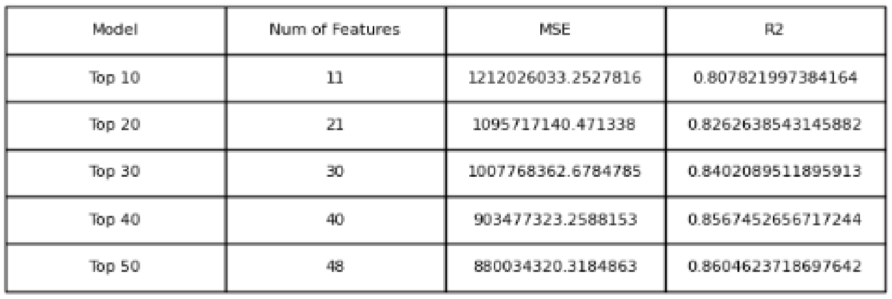
2.4.2 Ordinary least squares and evaluation

Using scikit-learn, I applied linear model to each of the feature sets created from the top 10, 20, 30, 40, and 50 highest MI values. Using these models, I predicted the label value of the training data and used this to calculate the MSE and R^2 score.

The results show that increasing the number of features increases the two scores on the training data (Figure 8). This is expected, because more features does increase the two metrics on the training data. Eventually I decided to use the Top 40 model, because this was where the R^2 score would no longer increase as much going from Top 40 to Top 50 model. I preferred using the R^2 score as an evaluation metric, because this metric is a relative value on the variance. However, using the MSE score does give a similar result of good decrease with increasing number of features until the Top 40 model, where MSE no longer decreases as much.

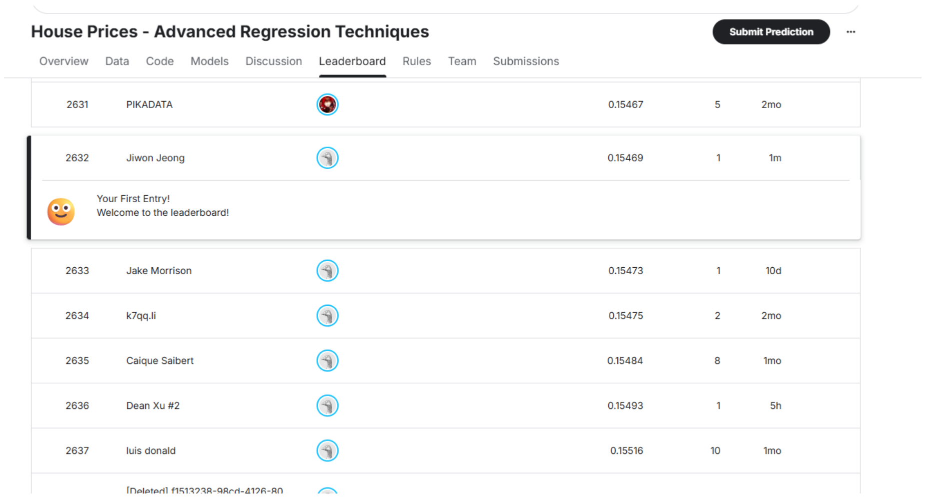
2.4.3 Applying the model to testing and takeaways

Lastly, I used the Top 40 model to predict the prices of the testing data and submitted to Kaggle (Figure 9). I'm happy with my results, and I learned a lot of new tools and techniques, especially in feature engineering and using frameworks. Also, I learned how to create a small pipeline to automate preprocessing.



Model	Num of Features	MSE	R2
Top 10	11	1212026033.2527816	0.807821997384164
Top 20	21	1095717140.471338	0.8262638543145882
Top 30	30	1007768362.6784785	0.8402089511895913
Top 40	40	903477323.2588153	0.8567452656717244
Top 50	48	880034320.3184863	0.8604623718697642

Figure 8: Evaluation metrics for models on training data



2631	PIKADATA	0.15467	5	2mo
2632	Jiwon Jeong	0.15469	1	1m
2633	Jake Morrison	0.15473	1	10d
2634	k7qq.li	0.15475	2	2mo
2635	Caique Saibert	0.15484	8	1mo
2636	Dean Xu #2	0.15493	1	5h
2637	luis donald	0.15516	10	1mo
	[Deleted] f1513238-98cd-4126-80			

Your First Entry!
Welcome to the leaderboard!

Figure 9: My position on Kaggle competition

Based on my leaderboard ranking, I was 2633/4034, so my predictions were somewhat below average. On reflection, I made a lot of ambiguous choices like picking smoothing coefficient, choosing how many top MI features, and interpreting correlation that I want to make more defined next time. Domain-wise, I realized that there is a huge importance to understanding your problem and data. I was able to deal with most (not all) missing values appropriately and did a few feature compositions. However, I think creative, relevant feature engineering is a big area for improvement.

3 Written Assignment

3.1 Q1

Supervised machine learning requires assumptions for several reasons. First, the point of ML is to determine the data generating curve/function between the input variable and the output label. Simply with this objective, we are assuming that such a data generating curve exists (and that the generation is not random).

Second, the data generating curve is not known so we must derive our prediction of what it is based on limited sample data. This is why we cannot just learn from data alone. Data alone does not span the entire data generating curve input space. However, when we use these samples, we are considering each data point as a randomly sampled (random variable) of the input. This creates an assumption that the attribute sample values are selected independently (from other values) and identically distributed (along the same distribution).

Third, we take the samples and extrapolate/interpolate the data generating curve through a mathematical model class. For example, we have used a linear model in linear regression. By selecting a specific model class, we are creating assumptions (like monotonicity in linear model class) of the relationship between the feature and label.

3.2 Q2

(see written page)

3.3 Q3

One explanation is that your model class is not appropriate for your dataset. For example, you might be trying to model the number of bacteria in a colony (label) with hours since incubation (feature). It is known that there is an exponential growth relationship, but if you try to use a linear model class, your model will have poor predictability and large differences (loss) between your model output and training actual label value. Basically, the model class mathematically cannot represent the exponential relationship well across the entire feature space.

Another explanation is that your features have a lot of variance due to random noise. For example, you are trying to model the concentration of dissolved copper (label) with the absorbance of some specific wavelength (feature). For very low absorbances, your instrumentation may not have good precision and the random noise will dominate the feature value. With significant variance, you are statistically unlikely to have enough data samples to determine the true data generative curve, leading to poor predictions. Simultaneously, your training loss will be high, because most loss functions calculates based around some absolute distance between the predicted label and actual label values. This will capture the variance in the feature that gets transformed into variance in the predicted label value by the model calculation. Also, in this specific scenario, if your absorbances are very low, your concentrations are also low and may also result in significant variance in the label values.

3.4 Q4

3.4.1 a

When users see two features of a linear model (or two variables of a linear multivariable equation), they will think that they can increase one feature (or variable) without increasing the other and control them independently. This is a result of (not directly is) the independence assumption of linear regression. This mental model breaks if we include features that must increase/decrease together. To maintain the mental model and interpretability, we should not include these features that have a relationship with other features.

However, keeping colinear variables will improve prediction. This is because

3.4.2 b

The assumption of monotonicity is invalidated. Higher body temperature results in higher mortality. So based on monotonicity, a low body temperature should be a very low mortality rate (compared to any other higher temperature). However, we see that going from normal to low body temperature results in a higher, not lower, mortality. To still use linear regression, we can apply a transformation on the body temperature feature to be the absolute difference from the median body temperature. So now we think of the feature as ”amount of temperature away from normal”.

3.4.3 c

The assumption of independence is invalidated. With independence, having ”western” is likely no spam means that no matter what other feature are, the feature of having ”western” does not contribute to spam categorization. Same thing with ”union”. So based on independence, we expect ”western union” to not be categorized as spam, however the opposite occurs.

We can still apply linear regression by combining the two features into one feature that considers if ”western union” is in the email or not. We would drop the individual ”western” and ”union” features.

② a) $J(\theta_0, \theta_1)$

$$= \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

$$= \sum_{i=1}^{n} \left(y^{(i)} - \theta_0 - \theta_1 x^{(i)}\right)^2$$

$$J(\theta_0, \theta_1) = \sum_{i=1}^{n} \left(y^{(i)^2} + \theta_0^2 + \theta_1^2 x^{(i)^2} - 2\theta_0 y^{(i)} - 2\theta_1 x^{(i)} y^{(i)} + 2\theta_0\theta_1 x^{(i)}\right)$$

$$\frac{\partial}{\partial\theta_0} J(\theta_0, \theta_1) = \frac{\partial}{\partial\theta_0} \sum_{i=1}^{n} \left(y^{(i)^2} + \theta_0^2 + \theta_1^2 x^{(i)^2} - 2\theta_0 y^{(i)} - 2\theta_1 x^{(i)} y^{(i)} + 2\theta_0\theta_1 x^{(i)}\right)$$

$$\frac{\partial}{\partial\theta_0} J(\theta_0, \theta_1) = \sum_{i=1}^{n} \frac{\partial}{\partial\theta_0} \left(y^{(i)^2} + \theta_0^2 + \theta_1^2 x^{(i)^2} - 2\theta_0 y^{(i)} - 2\theta_1 x^{(i)} y^{(i)} + 2\theta_0\theta_1 x^{(i)}\right)$$

$$= \sum_{i=1}^{n} \left(0 + 2\theta_0 + 0 - 2y^{(i)} - 0 + 2\theta_1 x^{(i)}\right)$$

$$\boxed{\frac{\partial}{\partial\theta_0} J(\theta_0, \theta_1) = 2n\theta_0 + \sum_{i=1}^{n} \left(-2y^{(i)} + 2\theta_1 x^{(i)}\right)}$$

$$\frac{\partial}{\partial\theta_1} J(\theta_0, \theta_1) = \frac{\partial}{\partial\theta_1} \sum_{i=1}^{n} \left(y^{(i)^2} + \theta_0^2 + \theta_1^2 x^{(i)^2} - 2\theta_0 y^{(i)} - 2\theta_1 x^{(i)} y^{(i)} + 2\theta_0\theta_1 x^{(i)}\right)$$

$$\frac{\partial}{\partial\theta_1} J(\theta_0, \theta_1) = \sum_{i=1}^{n} \frac{\partial}{\partial\theta_1} \left(y^{(i)^2} + \theta_0^2 + \theta_1^2 x^{(i)^2} - 2\theta_0 y^{(i)} - 2\theta_1 x^{(i)} y^{(i)} + 2\theta_0\theta_1 x^{(i)}\right)$$

$$= \sum_{i=1}^{n} \left(0 + 0 + 2\theta_1 x^{(i)^2} + 0 - 2x^{(i)} y^{(i)} + 2\theta_0 x^{(i)}\right)$$

$$\boxed{\frac{\partial}{\partial\theta_1} J(\theta_0, \theta_1) = \sum_{i=1}^{n} \left(2\theta_1 x^{(i)^2} - 2x^{(i)} y^{(i)} + 2\theta_0 x^{(i)}\right)}$$

b) $\frac{\partial}{\partial \theta_0} J(\theta_0^*, \theta_1) = 0$

$$2n\theta_0^* + \sum_{i=1}^{n} \left(-2y^{(i)} + 2\theta_1 x^{(i)}\right) = 0$$

$$\theta_0^* + \frac{1}{n}\sum_{i=1}^{n}\left(-y^{(i)} + \theta_1 x^{(i)}\right) = 0$$

$$\theta_0^* - \frac{1}{n}\sum_{i=1}^{n} y^{(i)} + \frac{\theta_1}{n}\sum_{i=1}^{n} x^{(i)} = 0$$

$$\theta_0^* - \bar{y} + \theta_1 \bar{x} = 0$$

$$\boxed{\theta_0^* = \bar{y} - \theta_1 \bar{x}}$$

$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1^*) = 0$

$$\sum_{i=1}^{n}\left(\cancel{2}\theta_1^* x^{(i)2} - \cancel{2}x^{(i)} y^{(i)} + \cancel{2}\theta_0^* x^{(i)}\right) = 0$$

$$= \theta_1^* \sum_{i=1}^{n} x^{(i)2} + \theta_0^* \sum_{i=1}^{n} x^{(i)} - \sum_{i=1}^{n} x^{(i)} y^{(i)} = 0$$

$$= \theta_1^* \sum_{i=1}^{n} x^{(i)2} + (\bar{y} - \theta_1^* \bar{x}) \sum_{i=1}^{n} x^{(i)} - \sum_{i=1}^{n} x^{(i)} y^{(i)} = 0$$

$$= \theta_1^* \sum_{i=1}^{n}\left(x^{(i)2} - \bar{x}\, x^{(i)}\right) = \sum_{i=1}^{n} x^{(i)} y^{(i)} - \bar{y}\, x^{(i)}$$

$$\boxed{\theta_1^* = \frac{\sum_{i=1}^{n} x^{(i)}\left(y^{(i)} - \bar{y}\right)}{\sum_{i=1}^{n} x^{(i)}\left(x^{(i)} - \bar{x}\right)}}$$

c) $\sum_{i=1}^{n} e^{(i)} = \sum_{i=1}^{n} (y^{(i)} - (\theta_0^* + \theta_1^* x^{(i)}))$

$$= \sum_{i=1}^{n} \left(y^{(i)} - \bar{y} - \theta_1 \bar{x} - \frac{\sum_{i=1}^{n} x^{(i)}(y^{(i)} - \bar{y})}{\sum_{i=1}^{n} x^{(i)}(x^{(i)} - \bar{x})} x^{(i)}\right)$$

$$= \sum_{i=1}^{n} \left(y^{(i)} - \frac{1}{n}\sum_{i=1}^{n} y^{(i)} - \theta^* \frac{1}{n}\sum_{i=1}^{n} x^{(i)} - \frac{\sum_{i=1}^{n} x^{(i)}(y^{(i)} - \bar{y})}{\sum_{i=1}^{n} x^{(i)}(x^{(i)} - \bar{x})} x^{(i)}\right)$$

$$= \sum_{i=1}^{n} y^{(i)} - \cancel{\frac{n}{n}} \sum_{i=1}^{n} y^{(i)} - \theta_1^* \cancel{\frac{n}{n}} \sum_{i=1}^{n} x^{(i)} - \theta_1^* \sum_{i=1}^{n} x^{(i)}$$

$$= 0 \quad - 0 = \boxed{0}$$

This means that if we have the optimal θ_0^*, θ_1^*, then there will be no summed difference between the real label and the modeled label. The model line will lie between the points so that all the residuals cancel out.