**Answers**

**2.1**

3.

I chose to drop the **‘date’** and **‘yr\_renovated’** features because both showed almost zero correlation with the price. Removing them simplifies the model without compromising performance.

I kept all the other features. While some of them had only a small correlation with price, I assumed that a well-optimized model would naturally assign them small weights if they’re not important.

In addition, I created three new features: **‘grade15’**, **‘view15’**, and **‘waterfront15’**. These are meant to reflect the overall quality of the neighborhood. Even though only houses directly on the waterfront or with a view get a positive value in the original features, it’s reasonable to assume that the surrounding area is also more expensive due to its proximity to these attributes. Similarly, **‘grade15’** gives an estimate of the neighborhood’s overall condition—areas with well-maintained homes tend to be more expensive, and it’s likely that nearby houses have similar characteristics.

For handling invalid or missing values in the training set, I tested two approaches. The first was correcting the data (e.g., setting **‘yr\_renovated’** to 0 when it appeared to be before **‘yr\_built’**), and the second was simply deleting the invalid rows. Since both approaches gave roughly the same results, I chose to delete the rows to keep things simple.

Regarding missing values in general, I split the data into training and test sets in a way that ensured all rows with missing values were placed in the training set. This way, I could handle (i.e., delete) them before training, without affecting the test set.

Finally, I set the **‘id’** column as the index of the dataset. This was for two reasons: first, the ID itself has no correlation with the price, so it should be removed as a feature; second, setting it as the index makes it easier to look up specific houses later on.

**6.**

As shown in the plot, the **mean squared error (MSE) on the test set decreases** as the percentage of the training set used increases. This trend is expected—using more training data typically improves the model's ability to generalize, especially when the number of samples is much larger than the number of features, which helps prevent overfitting.

Two additional patterns can be observed:

1. **The curve becomes smoother** as the training percentage increases.
2. **The confidence interval narrows** with larger training set sizes.

These trends reflect the fact that model estimates become more stable when trained on more data. When using a small portion of the training set, performance can vary greatly depending on which specific samples are selected. For example, if a sample by chance includes houses with typical feature values and no outliers, the model may perform unusually well. On the other hand, other random samples might contain less representative data, leading to worse predictions.

Because of this randomness, the MSE values are more volatile at lower percentages, resulting in larger confidence intervals and a less stable curve. As the training size increases, the sample becomes more representative of the full dataset, making the results more consistent and reducing the variance between different runs. This leads to tighter confidence intervals and a smoother loss curve.

**2.2**

3.

The scatter plot shows the average daily temperature in Israel as a function of the day of the year, with different colors representing different years. The overall shape is clearly seasonal—temperatures rise from the beginning of the year, peak around mid-year, and then drop again toward the end. This behavior is smooth and predictable, so a polynomial model seems like a good fit.

Looking at the curve, it seems that a **polynomial of degree 4 or 5** should be enough to capture the overall trend without overfitting. A lower degree wouldn’t fully represent the shape of the data, while a higher one might follow noise instead of the real pattern.

The bar plot shows the standard deviation of the daily temperatures in each month. We can see that:

* The **lowest standard deviations** are in **July and August**, which makes sense because the weather in summer is usually very stable.
* The **highest standard deviations** are in **March–April** and **November–December**, which are the transition months where the temperature can vary a lot from day to day.

If we train a polynomial model using random days from the year, it’s likely to **perform better during the summer**, since the temperatures are more stable and easier to predict. On the other hand, it will probably **struggle more in the spring and fall**, when the weather is less consistent.

So in short, the model won’t perform equally across all months—some periods (especially summer) are easier to predict than others.

4.

From the graph, we can see that:

* **Israel and Jordan** show **very similar seasonal patterns**. Both countries have a clear rise in temperature from winter to summer, peaking around July–August, and dropping again toward December. The overall shape and temperature range are almost the same.  
   → A model trained on Israel’s data will likely **work well for Jordan too**, since their climate behavior is very close.
* **The Netherlands** has a similar seasonal structure (colder in winter, warmer in summer), but the **temperature range is lower**, and the **winter is much colder** compared to Israel. While the general shape is the same, the values are quite different.  
   → The model might **capture the trend**, but will likely **underestimate or overestimate the actual temperatures**.
* **South Africa** has a completely **opposite pattern**—its coolest months are around July–August, and warmest around January–February. This makes sense since it's in the Southern Hemisphere.  
   → A model trained on Israel’s data would **not work at all for South Africa**, since the pattern is flipped.

In summary, the model fitted for Israel could probably generalize well to **Jordan**, might **partially work for the Netherlands**, but would **fail for South Africa** due to the opposite seasonal trend.

5.

The bar plot and the test losses for each degree:

The lowest test loss was achieved with **k = 5**, with a loss of **3.22**. This means the degree-5 polynomial fits the data best on the test set. Even though degrees 4 and 3 had losses very close to that, we picked **k = 5** as the best fit, since it gave the lowest error.

We can also see that using a higher degree beyond 5 actually **hurts performance**, with the loss increasing significantly from **k = 6** onward—most likely due to **overfitting**. This is a good example of how increasing model complexity doesn’t always lead to better generalization.

The bar plot shows the test loss of the model trained on Israel when applied to Jordan, the Netherlands, and South Africa.

We can see that:

* The model **performs best on Jordan**, which makes sense because its temperature pattern is very similar to Israel’s.
* The loss is **higher for the Netherlands**, which has a similar seasonal shape but different temperature ranges, especially in winter.
* The **worst performance is on South Africa**, where the temperature pattern is reversed due to it being in the southern hemisphere.

In short, the model works well only for countries with similar climate trends to Israel.