Task 1

Point A

Observations in X_test.csv.gz: 670

Variables in X_test.csv.gz: 9000

Is table missing any data? No

Observations in X_train.csv.gz: 3794

Variables in X_train.csv.gz: 9000

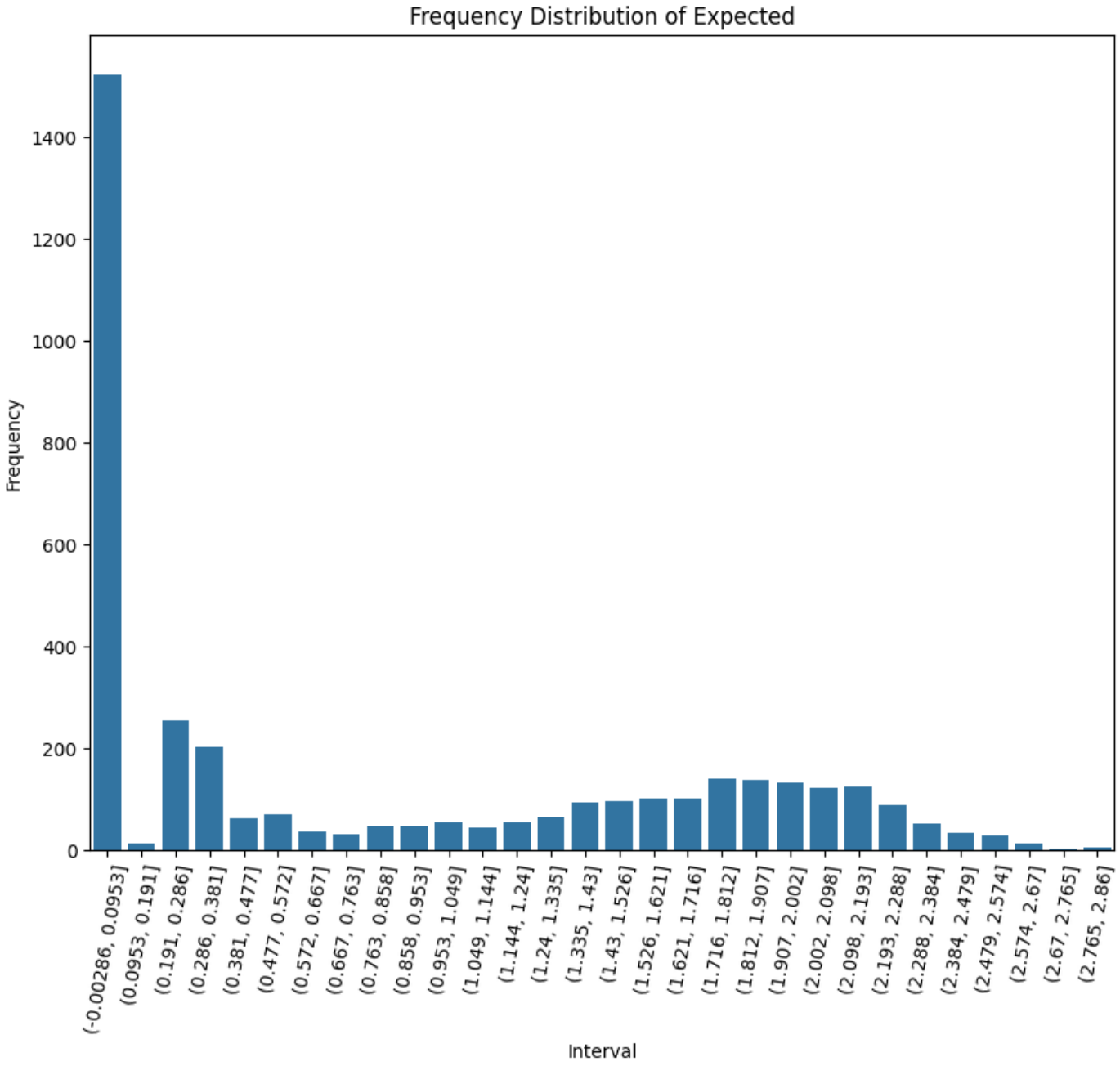
Is table missing any data? No

Observations in y_train.csv.gz: 3794

Variables in y_train.csv.gz: 2

Is table missing any data? No

Point B



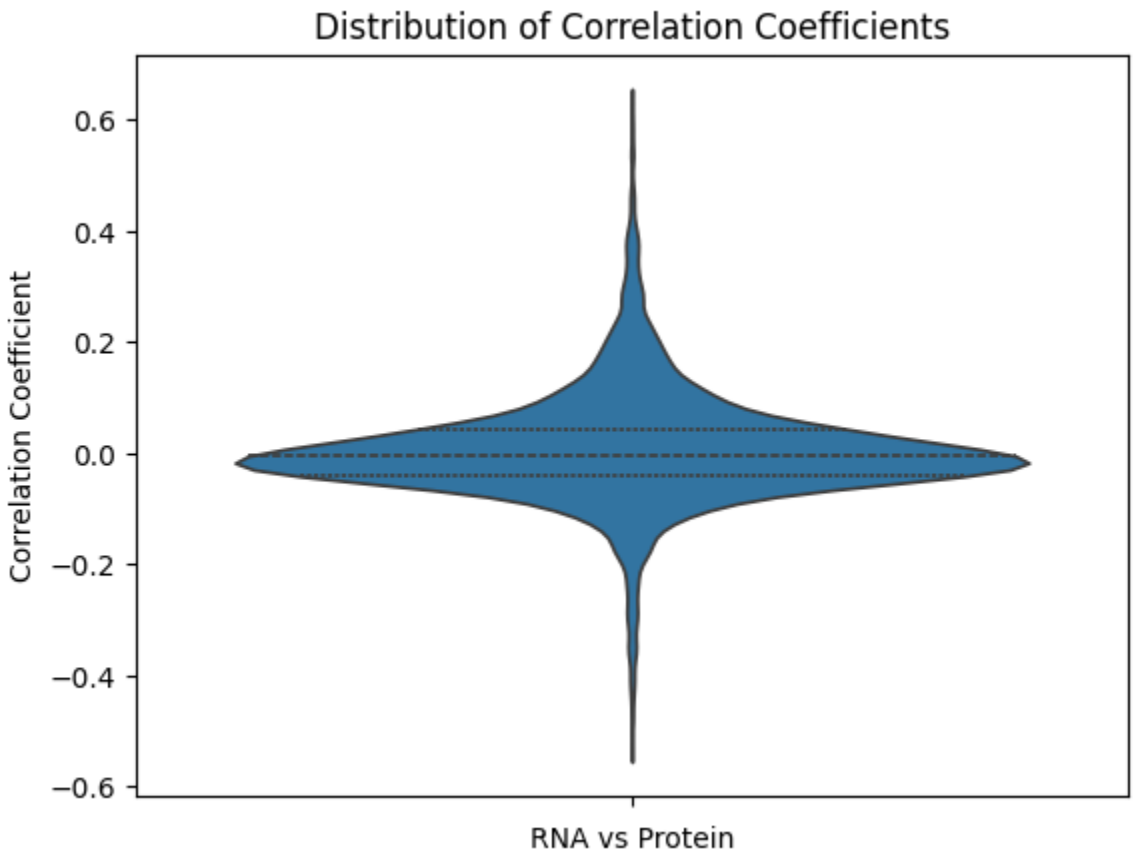
I computed a histogram and it's also available in the notebook. I have also displayed descriptive statistics, which have given me even more insight.

```
count    3794.000000
mean        0.791096
std         0.860856
min         0.000000
25%         0.000000
50%         0.311748
75%         1.662514
max         2.860416
Name: Expected, dtype: float64
Zero appears 1523 times in the provided data.
Values below 0.14: 1523
```

There are really many observations that are equal to zero. Then there is a jump and there are no data points between 0 and 0.14. Then the data follows some distribution with 2 distint peaks around 0.25 and

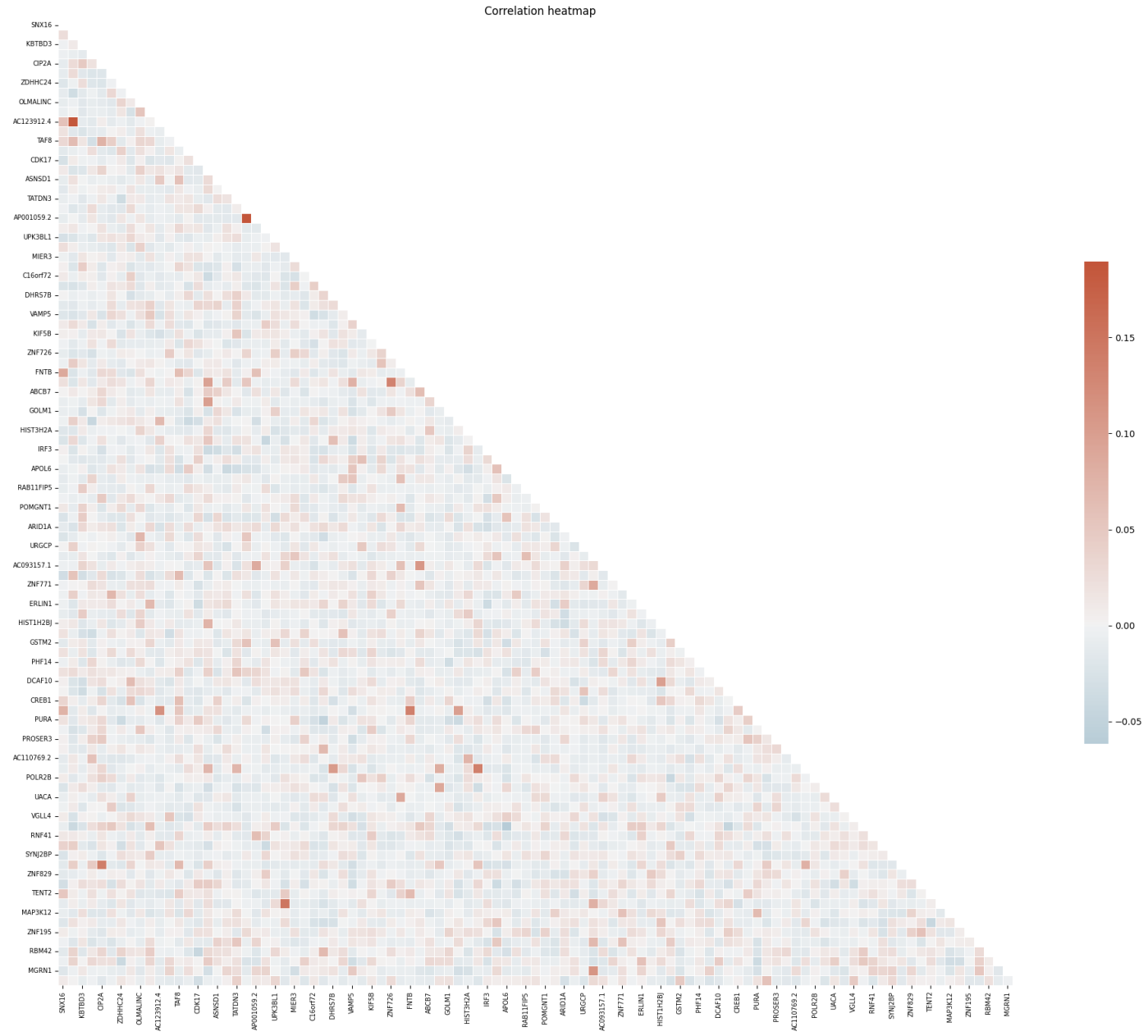
1.8.

Point C



Most of the correlations are not that meaningful and sit between -0.1 and 0.1. But there is a significant number of strong correlations ranging up to 0.6 in absolute value.

One could assume this pattern is typical for high-dimensional biological datasets. It would make sense that only a subset of features (genes) are strongly associated with the target variable.



As seen on heatmap variables also correlate with themselves. There are about 20 correlation coefficients in range from 0.1 and above.

These correlations might represent gene co-expression patterns (genes involved in similar pathways or processes).

Task 2

ElasticNet Model

ElasticNet is a linear regression model that combines **Lasso Regression** (L1 reg) and **Ridge Regression** (L2 reg). It is effective for datasets with many predictors, especially when some predictors are either irrelevant or highly correlated. L1 regularization facilitates **variable selection** while L2 helps with **regularization** simultaneously, improving model interpretability and preventing overfitting.

Optimization Function

The ElasticNet tries to minimize the following function:

$$\frac{1}{2n}\|y - X\beta\|_2^2 + \alpha\left[\lambda\|\beta\|_1 + \frac{1}{2}(1-\lambda)\|\beta\|_2^2\right]$$

source: scikit-learn.org

Where:

- y : Target variable.
- X : Input features.
- β : Model coeffs.
- $\|...\|_1$: L1 norm.
- $\|...\|_2^2$: L2 norm.
- α : Regularization strength.
- λ: L1 ratio.

Parameters Estimated

1. **Regression coefficients** β
2. **Intercept** β_0 **:**
    - The constant term.

Hyperparameters

1. α (Regularization strength):

- Controls the overall strength of regularization.

2. λ (L1 ratio):

- Determines the balance between L1 and L2 regularization
- Special case: $\lambda = 1$**: Lasso Regression**
    - Pure L1 regularization.
    - Performs **feature selection** by shrinking some coefficients to exactly zero.
    - Struggles with multicollinearity.
- Special case: $\lambda = 0$**: Ridge Regression**
    - Pure L2 regularization.
    - Handles **multicollinearity** by stabilizing correlated coefficients.
    - Does not perform feature selection; all coefficients remain non-zero.

By tuning α and λ ElasticNet adapts to the specific properties of the dataset.

Why did i choose 5-fold validation:

- Number of Observations:
    - There is 3,794 observations in training data.
    - Splitting the data into 5 folds means each fold will have ~759 observations for validation and ~3035 for training.
    - This should be sufficient for training models.
- Dimensionality of Data:
    - The data has 9,000 variables, which makes it high-dimensional.
    - With 5 folds, the training set hopefully remains large enough to avoid overfitting.
- Computational cost:
    - Training Random Forests is computationally demanding, so choosing small enough fold number is important.
    - Reduces variance in validation compared to 3 folds, while avoiding the computational cost of 10 or more folds.

First conclusions:

Model doesn't seem to converge when l1_ratio is 0, but it gets the lowest errors in that case. It may be beneficial to look for l1_ratio close to 0.

1st pass results:

MSE loss:
[[0.12633889 0.11695261 0.12696826 0.13579096 0.20370874]
[0.11642005 0.13681605 0.16317229 0.18103404 0.33775193]
[0.11732301 0.15469409 0.18932172 0.22007956 0.44057694]
[0.11971264 0.17062719 0.2144733 0.25460069 0.59316338]
[0.12246912 0.18431365 0.23853962 0.2880758 0.73607446]]

lambdas:
[[0. 0.05 0.125 0.2 1.]
[0. 0.05 0.125 0.2 1.]
[0. 0.05 0.125 0.2 1.]
[0. 0.05 0.125 0.2 1.]
[0. 0.05 0.125 0.2 1.]]

alphas:
[[0.1 0.1 0.1 0.1 0.1]
[0.325 0.325 0.325 0.325 0.325]
[0.55 0.55 0.55 0.55 0.55]
[0.775 0.775 0.775 0.775 0.775]
[1. 1. 1. 1. 1.]]

Min MSE loss: 0.11642005435572308 for alpha = 0 and lambda = 0.325

Insights

- Small lambdas (L1 Ratio):
    - When lambda is close to 0 (more Ridge-like behavior), MSE tends to be smaller, particularly for smaller values of alpha (weaker regularization).
    - Best performance occurs at (alpha=0.325, lambda=0), where MSE = 0.1164.
- High lambdas (L1 Ratio):
    - As lambda increases to 1 (purely Lasso behavior), the MSE increases significantly, indicating poor performance for higher regularization strength.
- Increasing alphas:
    - Larger values of alpha (stronger regularization) generally lead to higher MSE, particularly for high lambda.

Task 3

MSE loss:
[[[0.12834956 0.11554959]
[0.12631654 0.11440359]]
[[0.1260375 0.11304446]
[0.12462456 0.1120295]]]

n_estimators:
[[[60 60]
[100 100]]
[[60 60]
[100 100]]]

max_depth:
[[[None None]
[None None]]
[[30 30]
[30 30]]]

max_features:
[[['sqrt' None]
['sqrt' None]]
[['sqrt' None]
['sqrt' None]]]

Insights:

- max_features (sqrt vs None):
    - When max_features = None, the MSE is consistently lower compared to max_features = 'sqrt', suggesting that allowing all features at each split improves performance.
- n_estimators (60 vs 100):
    - More trees (from n_estimators = 60 to n_estimators = 100) reduce MSE for both max_features

= 'sqrt' and max_features = None.

- max_depth (None vs 30):
  - Limiting the depth (max_depth = 30) generally leads to better performance, it may prevent overfitting.
- Optimal Configuration:
  - The lowest MSE occurs when n_estimators = 100, max_depth = 30, and max_features = None: MSE = 0.1120

Summary

The table below compares the **MSE validation loss** for ElasticNet, Random Forest, and the reference model. Cross-validation was performed using the same 5 divisions to ensure consistency in evaluation.

Model	Hyperparameters	MSE Validation Loss	Notes
ElasticNet	$\alpha \in \{0.1, 0.325, 0.55, 0.775, 1.0\}$ $\lambda \in \{0, 0.05, 0.125, 0.2, 1\}$	Best: **0.1164**	Best performance occurs for $\alpha = 0.325$ and $\lambda = 0$ (more Ridge-like behavior).
Random Forest	$n\_estimators \in \{60, 100\}$, $max\_depth \in \{None, 30\}$, $max\_features \in \{\text{sqrt}, \text{None}\}$	Best: **0.1120**	Best performance with $n\_estimators = 100$, $max\_depth = 30$, $max\_features = None$.
Reference	None	**0.7401**	Baseline model for comparison; assigns $\hat{y} = \bar{y}$ to all predictions.

Key Observations

1. **ElasticNet Performance**:
   - Best MSE: **0.1164**, achieved when $\alpha = 0.325$ and $\lambda = 0$.
   - ElasticNet performs better with smaller λ (more Ridge-like regularization).
2. **Random Forest Performance**:
   - Best MSE: **0.1120**, achieved when $n\_estimators = 100$, $max\_depth = 30$, and $max\_features = None$.
   - Adding more trees (higher $n\_estimators$) consistently improves performance, and restricting max_depth to 30 may reduce overfitting.
3. **Reference Model**:
   - MSE: **0.7401**, much higher than ElasticNet or Random Forest.

Best Model

The **Random Forest** model is the best based on the cross-validation results:

- **Lowest MSE**: Achieved a validation loss of **0.1120**, outperforming other models.
- **Robustness**: Random Forest can model non-linear relationships, which may have contributed to its performance.

Conclusion

Random Forest is the best-performing model in this comparison. The improvement justifies the added complexity of Random Forest over ElasticNet.

Task 4

I played around with a couple of methods, but came to the conclusion that Random forest method is the most robust one in this scenario. I think it comes down to the fact that we have really high dimensionality, but not so many observations.

I ran many iterations doing grid searches for best hyperparemeters and came to this result:

model = RandomForestRegressor(n_estimators = 500, max_depth = 37, max_features = 0.1,)

which gets around 0.32 RMSE in 5-fold cross-validation, which i think is respectable.

- n_estimators The number of decision trees in the forest. More trees generally improve performance by reducing variance.
- max_features
    - Controls the number of features randomly considered at each split.
    - 0.1 is a healthy performance / computation time balance.
- max_depth
    - Limits the depth of each tree.
    - Can prevent overfitting.
- min_samples_split
    - The minimum number of samples required to split a node.
    - Larger values force trees to be shallower.
- min_samples_leaf
    - The minimum number of samples a leaf node must have.
    - Prevents very small leaf nodes.

I tried playing with min_samples_split and min_samples_leaf, but came to the conclusion that leaving default values gives the best results.

I also tried zeroing out small values to fit the distribution better, but it didn't help that much.

I put more emphasis on observations resulting in a smaller value.