Feature Analysis Techniques for Disease Prediction from Protein Isoforms

1 Introduction

Feature analysis is a vital step in machine learning (ML) to identify which features (input variables) most influence a model's ability to predict a target variable. In this context, the dataset contains protein isoform data with two features: `p-value_lowest` (statistical significance of the isoform) and `effector_score` (a measure of the isoform's biological effect). The target is to predict a disease-related outcome (e.g., disease severity or risk) based on these protein characteristics. This document explains four feature analysis techniques—Built-in Feature Importance, Permutation Importance, Linear Regression Coefficient Analysis, and Feature-Target Correlation Analysis—in a beginner-friendly way with technical details, tailored to the disease prediction task.

2 Built-in Feature Importance Analysis

2.1 What is it?

Built-in feature importance is a method used by tree-based ML models (e.g., Random Forests, Gradient Boosting) to quantify how much each feature, such as `p-value_lowest` or `effector_score`, contributes to predicting disease from protein isoforms. It relies on internal metrics, like how often a feature is used to split data in decision trees or how much it reduces impurity (e.g., Gini index).

2.2 Why use it?

This method is fast and specific to tree-based models, revealing which protein features drive disease predictions. It's useful for:

- Identifying key protein properties linked to disease.
- Simplifying models by removing less important features.
- Explaining model predictions to biologists or clinicians.

2.3 How does it work?

1. **Extract Importance Scores**: The `feature_importances_` attribute provides scores based on how much each feature (e.g., `effector_score`) reduces impurity during training.
2. **Normalize to Percentages**: Convert raw scores to percentages by dividing each by the sum of all scores and multiplying by 100.
3. **Include Weights**: Incorporate statistical weights (e.g., normalized p-values from `weights_norm`) to contextualize importance with significance.
4. **Store and Display**: Organize results in a table and display the top features.

2.4 Technical Details

- **Models**: Works with models like Random Forest or XGBoost, where `feature_importances_` is available.
- **Formula**: For a feature f, importance is based on the total reduction in impurity across all splits involving f. Percentage importance is:

$$\text{Importance}_{\%} = \left(\frac{\text{Importance}_f}{\sum \text{Importance}_i} \right) \times 100$$

- **Assumptions**: Assumes the model is trained on features `p-value_lowest` and `effector_score`, aligned with `feature_names_filtered`.

2.5 Example Output

For a Random Forest model predicting disease:

Feature	Imp. (%)	P-val. Wt.
`effector_score`	70.40	0.95
`p-value_lowest`	29.60	0.90

This suggests `effector_score` is the dominant feature for disease prediction, contributing 70.4% to the model's decisions.

2.6 Limitations

- Only applies to tree-based models.
- May overemphasize features used frequently in splits, even if they have limited predictive power for disease.

3 Permutation Importance Analysis

3.1 What is it?

Permutation importance measures a feature's importance by evaluating how much a model's performance (e.g., predicting disease risk) degrades when the feature's values (e.g., `effector_score`) are randomly shuffled. A large performance drop indicates high importance.

3.2 Why use it?

This model-agnostic method is robust and works with any model, making it ideal for validating feature importance in protein-disease studies. It's useful for:

- Confirming which protein features impact disease predictions.
- Detecting overfitting or data leakage in biological data.
- Comparing feature importance across models.

3.3 How does it work?

1. **Calculate Importance**: Randomly shuffle a feature's values in the test set and measure the change in model performance (e.g., negative mean squared error for disease severity).
2. **Repeat and Average**: Shuffle multiple times (e.g., 10) to compute mean importance and standard deviation.

3. **Normalize to Percentages**: Convert positive importance scores to percentages, setting negative or zero scores to 0%.
4. **Evaluate Significance**: Flag features as significant if their importance exceeds 2 standard deviations from zero.
5. **Flag Issues**: Identify features with high importance but low p-value weight (e.g., < 0.5) as potential signs of overfitting.

3.4 Technical Details

- **Function**: Uses scikit-learn's `permutation_importance` with `n_repeats=10`, `random_state=42`, and `scoring='neg_mean_squared_error'`.
- **Formula**: Importance for feature f:

  $$\text{Importance}_f = \text{Score}_{\text{original}} - \text{Score}_{\text{shuffled}}$$

  Percentage importance (for positive Importance_f):

  $$\text{Importance}_{\%} = \left(\frac{\text{Importance}_f}{\sum \text{Importance}_i} \right) \times 100$$

- **Significance**: A feature is significant if $|\text{Importance}_f| > 2 \times \text{Std}_f$.

3.5 Example Output

For a model predicting disease:

Feature	Raw Imp.	Imp. (%)	Sig.	P-val. Wt.
`effector_score`	1.234567	75.20	True	0.95
`p-value_lowest`	0.406789	24.80	True	0.90

This indicates `effector_score` strongly impacts disease prediction accuracy.

3.6 Limitations

- Computationally intensive due to repeated shuffling.
- Assumes test data (`X_test_filtered`) is representative of disease cases.
- Negative importance scores can complicate interpretation.

4 Linear Regression Coefficient Analysis

4.1 What is it?

This method uses the coefficients of a Linear Regression model to measure the importance of features like `p-value_lowest` and `effector_score` for predicting disease. The magnitude of a coefficient reflects the feature's linear impact on the disease outcome.

4.2 Why use it?

It's specific to linear models and interpretable for linear relationships in protein-disease data. Useful for:

- Understanding linear effects of protein features on disease.
- Comparing with non-linear methods (e.g., tree-based importance).
- Guiding feature selection for linear disease prediction models.

4.3 How does it work?

1. **Standardize Features**: Scale features to mean 0 and standard deviation 1 using `StandardScaler`.
2. **Fit Model**: Train a Linear Regression model on standardized data.
3. **Extract Coefficients**: Use absolute coefficients as importance measures.
4. **Normalize to Percentages**: Convert absolute coefficients to percentages.
5. **Include Weights**: Add statistical weights (e.g., p-values) for context.

4.4 Technical Details

- **Standardization**: Transforms features:

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

- **Model**: Linear Regression solves:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

  where β_1 and β_2 are coefficients for `p-value_lowest` and `effector_score`.

- **Importance**: Absolute coefficient percentage:

$$\text{Importance}_{\%} = \left(\frac{|\beta_i|}{\sum |\beta_j|} \right) \times 100$$

4.5 Example Output

Feature	Coef.	Imp. (%)	P-val. Wt.
`effector_score`	-2.1234	68.30	0.95
`p-value_lowest`	0.9876	31.70	0.90

This shows `effector_score` has a strong negative linear effect on disease prediction.

4.6 Limitations

- Assumes linear relationships, which may not capture complex disease-protein interactions.
- Sensitive to multicollinearity between `p-value_lowest` and `effector_score`.
- Requires standardization for fair comparison.

5 Feature-Target Correlation Analysis

5.1 What is it?

This method calculates the Pearson correlation coefficient between each feature (`p-value_lowest`, `effector_score`) and the disease outcome to measure their linear relationship strength.

5.2 Why use it?

It's model-agnostic and simple, revealing which protein features are linearly related to disease. Useful for:

- Identifying features for linear disease prediction models.
- Exploring relationships in protein-disease data.
- Detecting potential multicollinearity.

5.3 How does it work?

1. **Calculate Correlations**: Compute Pearson correlation between each feature and the disease outcome.
2. **Handle NaN**: Set invalid correlations (e.g., from constant features) to 0.
3. **Normalize to Percentages**: Convert absolute correlations to percentages.
4. **Display Top Features**: Show the top features by absolute correlation.

5.4 Technical Details

- **Pearson Correlation**: For feature x and target y:

$$\text{corr}(x, y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

- **Percentage**: Absolute correlation percentage:

$$\text{Correlation}_{\%} = \left(\frac{|\text{corr}(x, y)|}{\sum |\text{corr}(x_i, y)|} \right) \times 100$$

5.5 Example Output

Feature	Corr.	Corr. (%)
`effector_score`	-0.8500	67.50
`p-value_lowest`	0.4100	32.50

This suggests `effector_score` has a strong negative linear relationship with disease.

5.6 Limitations

- Only captures linear relationships, missing non-linear disease-protein interactions.
- Sensitive to outliers in `effector_score` or disease data.
- Doesn't account for feature interactions.

6 Comparing the Techniques

Technique	Model-Agn.	Non-Linear	Comp. Cost	Key Metric
Built-in Importance	No	Yes	Low	Impurity Red.
Permutation Importance	Yes	Yes	High	Perf. Drop
LR Coefficient	No	No	Medium	Coef. Mag.
Correlation Analysis	Yes	No	Low	Pearson Corr.

7 Conclusion

These four techniques provide complementary insights into the protein isoform dataset for disease prediction:

- **Built-in**: Quick for tree-based models, highlighting `effector_score` as critical for disease.
- **Permutation**: Robust validation, confirming key protein features across models.

- **LR Coefficient**: Reveals linear effects, useful for linear disease models.
- **Correlation**: Simple exploration of linear relationships in protein-disease data.

By combining these methods, researchers can identify critical protein features, guide disease prediction models, and ensure robust biological insights.