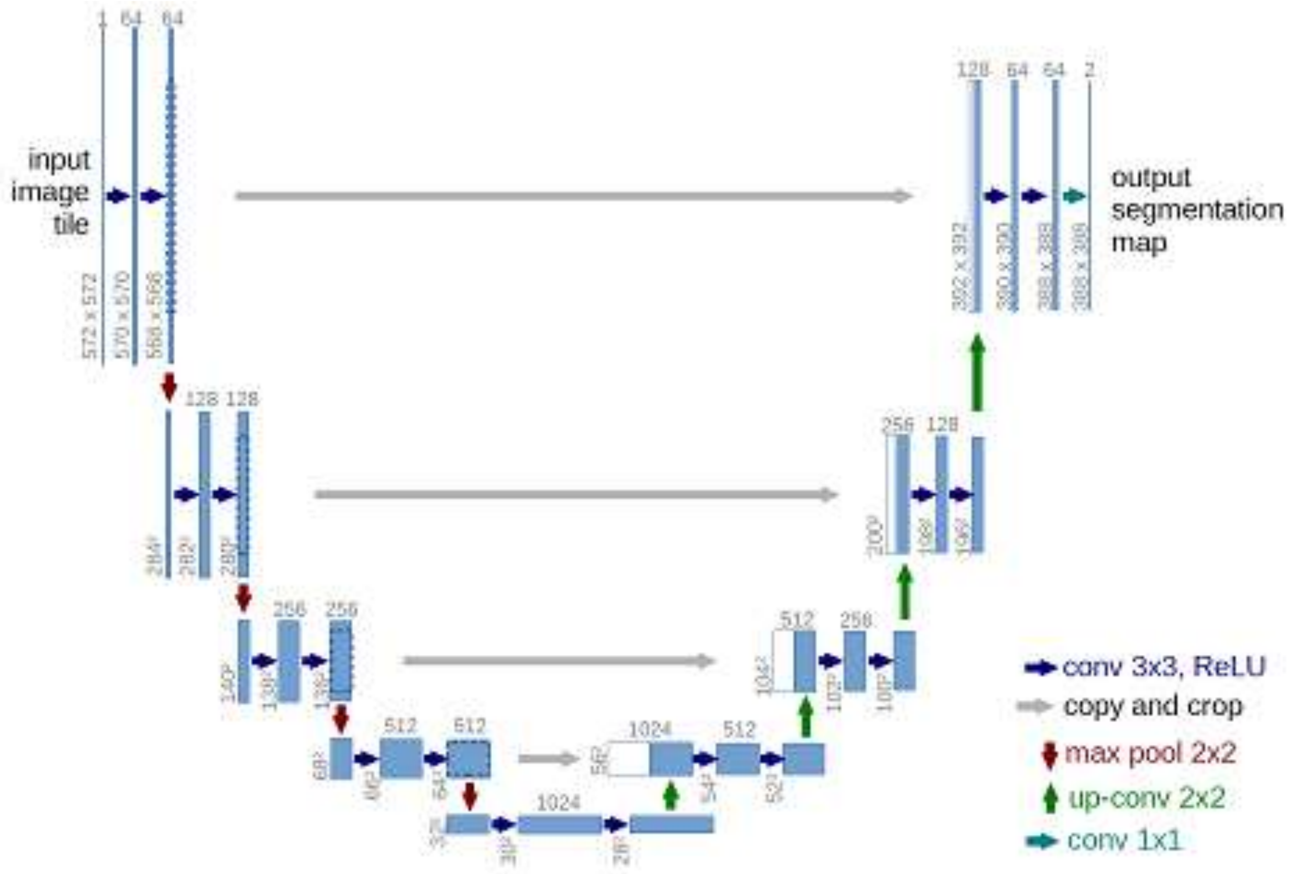
ML-Systems-Project-Phase2

1. **Choosing the right model candidates is crucial for the success of an ML project. Here are some tips to help us choose the right candidate models:**
    - Start with simple models: It's always a good idea to start with simple models to establish a baseline performance.
    - Consider the problem type: The type of problem we are trying to solve can help us narrow down the candidate models. In our project, the type of problem is generally image segmentation.
    - Evaluate the data: Analyze our data and see if it has any specific characteristics such as high dimensionality, sparsity, or class imbalance. Depending on the data, some models may perform better than others. In this project, we did this part in the first phase, and according to the limited number of images we have, we did some preprocessing.
    - Think about the desired output: Consider what kind of output we want from our model. For instance, in this project our output is image masks, which show membrane segmentation.
    - Consider computational limitations: Finally, consider the computational resources available to us, such as the amount of memory, processing power, and time we have. Respect to characteristics of our problem, which is image segmentation, we required high throughput computational resources, so we utilized google Colab pro.
    - Once we have a list of candidate models, we can use techniques such as cross-validation to evaluate their performance and select the best one. Ensemble learning can also be a useful technique to improve the performance of our model by combining the predictions of multiple models. But in our project we did not do any of these, because we could find a model from a recent research which was so close to our goal, therefore we just use the modification of that model.
    - In below, we will mention a list of models that are suitable for our task which is image segmentation for cell membranes, for cell membrane image segmentation, some of the commonly used models in the literature include:
    - **U-Net:** U-Net is a convolutional neural network (CNN) architecture that is commonly used for image segmentation tasks. It has a U-shaped architecture that consists of a contracting path and an expanding path, allowing it to capture both local and global features in the image. We only use this model as our main model, because after implementing a U-net we saw that our problem is solved. For our task the model architecture is like below. Also Loss function for the training is basically just a binary cross entropy. This deep neural network is implemented with Keras functional API, which makes it extremely easy to experiment with different interesting architectures. Output from the network is a 512*512 which represents mask that should be learned. Sigmoid activation function makes sure that mask pixels are in [0, 1] range. The model is trained for 5 epochs. After 5 epochs, calculated accuracy is about 0.97.

Also, we perform hyper parameter optimization for this task, and we save our final model with best parameters for evaluation part. (All the codes are available in the attached notebook.)



- **Mask R-CNN:** Mask R-CNN is a CNN architecture that extends the Faster R-CNN object detection model to perform instance segmentation. It can segment objects at pixel-level accuracy and has been used for various medical image segmentation tasks.
- **DeepLab v3+:** DeepLab v3+ is a CNN architecture that uses atrous convolution (also known as dilated convolution) to capture multi-scale features in the image. It has achieved state-of-the-art performance on various image segmentation benchmarks.
- **FCN:** Fully Convolutional Networks (FCN) are a type of CNN architecture that can be used for dense pixel-wise prediction tasks such as image segmentation. It was one of the first CNN architectures designed for semantic segmentation.
- **SegNet:** SegNet is another CNN architecture that is commonly used for image segmentation tasks. It has a similar architecture to U-Net but with a different approach to up sampling the feature maps.

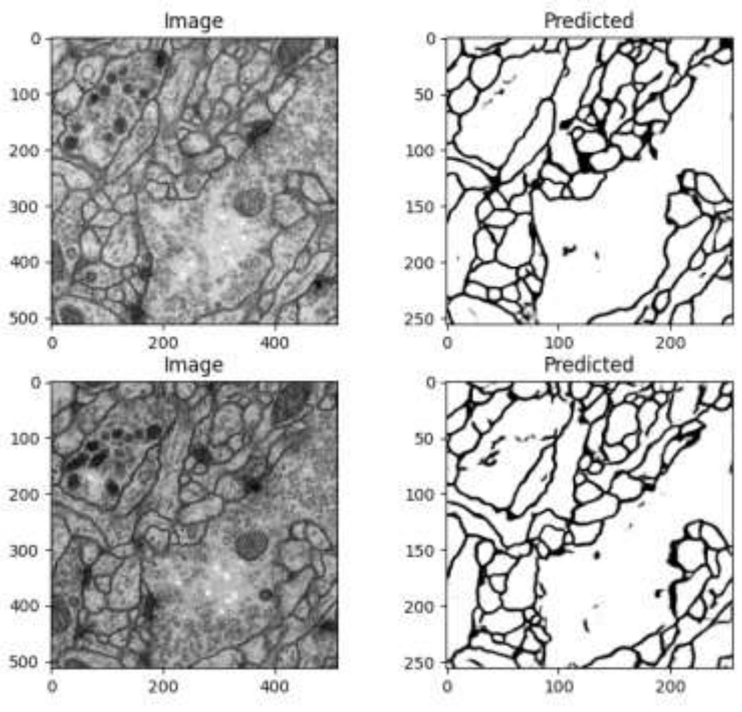
- These models can be trained on annotated data to learn to segment the cell membrane in the image. It's important to note that the performance of these models can vary depending on the quality and quantity of the training data, the preprocessing techniques used, and the hyper-parameters selected during training. Therefore, it's important to experiment with different models and hyper-parameters to find the best performing model for our specific task.
- In addition to the models mentioned above, there are also some specialized architectures and techniques that have been developed specifically for medical image segmentation tasks, such as attention-based models, multi-path networks, and adversarial training. These may also be worth exploring for our project depending on our specific needs.

2. **Choosing the right evaluation metric is crucial to assess the performance of our model accurately. For cell membrane image segmentation, some common evaluation metrics are:**
   - **Intersection over Union (IoU):** IoU is a popular metric for evaluating image segmentation models. It measures the overlap between the predicted segmentation and the ground truth segmentation. The IoU score ranges from 0 to 1, with higher values indicating better segmentation accuracy. With respect to our trained model, we get an IoU of 0.96.
   - **Dice coefficient:** The Dice coefficient is another popular metric for evaluating image segmentation models. It is similar to IoU and measures the overlap between the predicted and ground truth segmentation. The Dice coefficient ranges from 0 to 1, with higher values indicating better segmentation accuracy. We did not implement this evaluation metric.
   - **Precision, Recall, and F1-score:** Precision measures the fraction of true positive predictions among all positive predictions. Recall measures the fraction of true positive predictions among all actual positives. The F1-score is the harmonic mean of precision and recall, and it combines both metrics into a single value. We did not implement this evaluation metric.
   - **Mean Average Precision (mAP):** mAP is a popular metric for evaluating object detection models, but it can also be used for image segmentation. It measures the precision and recall of the model at different thresholds and computes the average precision across all thresholds. We did not implement this evaluation metric.
   - **Pixel Accuracy:** Pixel accuracy measures the percentage of pixels that are correctly classified by the model. This measure is a little tricky, but even in this metric we get a accuracy around 50 percent.

- It's important to choose a metric that is appropriate for our specific problem and aligns with our project goals. For instance, if we are interested in segmenting the cell membrane accurately, IoU or Dice coefficient may be more appropriate metrics as they measure the overlap between the predicted and ground truth segmentation. On the other hand, if we are interested in identifying all the cells in an image, mAP may be a more suitable metric.
- Once we have selected an appropriate metric, it's important to define how we will evaluate the model. One approach is to use a validation set to evaluate the model during training and select the best performing model based on the chosen metric. Another approach is to use cross-validation to evaluate the performance of the model on multiple folds of the data. Our selected approach is using test set. At first, we split our augmented data into three parts, train, validation, test. We used train and validation test for model training and hyper parameter optimization, and we use test set for our final evaluation.
- It's also important to set a baseline for our evaluation. The baseline can be a simple heuristic-based approach or a previously published result on a similar dataset. This baseline can help we assess the performance of our model relative to other approaches and determine if our model is performing well enough to be useful in practice. Our baseline is available in the main notebook, in the part of evaluation without ground truth.
- Finally, it's important to check the calibration of our model if needed. Calibration refers to the agreement between the predicted probabilities and the actual probabilities of the events. If our model is not well-calibrated, it may overestimate or underestimate the probabilities, leading to incorrect predictions. we can use techniques such as calibration plots or the Brier score to assess the calibration of our model. We did not do anything about this part.

3. **It's important to have a systematic approach to the training process to ensure that we are optimizing our model effectively. Here are some steps we can follow:**
    - Create a manual simple Artifact: Before we start training our models, it's important to create an artifact that documents our experimental setup, including the hyper-parameters, evaluation metrics, and any other relevant details. This artifact can help we keep track of our experiments and ensure that we can reproduce our results later. We can use a simple spreadsheet or a more sophisticated tool like MLflow to create this artifact.
    - Train our candidate models: Once we have selected our candidate models and defined our evaluation metrics, we can start training our models. It's a good idea to start with simpler models and gradually move towards more complex models to ensure that we are not overfitting the data. For this part, we start with a simple U-net model, with less amount of layers, and no dropouts, at first we got low accuracies, but our main model which was mentioned before reach a high accuracy.
    - Find the right hyper-parameters: Hyper-parameters are parameters that are set before training the model, such as learning rate, batch size, and regularization strength. Finding the right hyper-parameters is crucial for achieving good performance on our task. There are various techniques we can use to find the right hyper-parameters, such as grid search, random search, or Bayesian optimization. It's a good idea to use a validation set to evaluate the performance of the model with different hyper-parameters and select the best performing one. This part is available in the section of hyper parameter optimization in the notebook.
    - Save the results of our experiments: It's important to save the results of our experiments, including the evaluation metrics, hyper-parameters, and any other relevant details. We can save these results in our manual artifact, along with any relevant notes or observations. It's also a good idea to save the trained models and their weights so that we can use them later for inference or further training.
    - Regularly validate our model: During training, it's important to regularly validate the model on a separate validation set to ensure that it is not overfitting to the training data. We can use the chosen evaluation metric to assess the performance of the model on the validation set and monitor its progress over time.
    - Regularize our model: Regularization is an important technique to prevent overfitting and improve the generalization performance of our model. There are various regularization techniques we can use, such as L1 and L2 regularization, dropout, and data augmentation. It's a good idea to experiment with different regularization techniques to find the one that works best for our specific problem.
    - Use early stopping: Early stopping is another technique to prevent overfitting and improve the generalization performance of our model. It involves monitoring the performance of the model on the validation set and stopping the training process when the performance starts to deteriorate. This can help we avoid wasting computational resources and prevent the model from overfitting to the training data.

4. **If we do not have access to ground truth masks, we can still evaluate the performance of our model using visual inspection and qualitative analysis. Here are some steps we can follow:**
   - **Visualize the predicted masks:** We can use tools like Matplotlib or OpenCV to visualize the predicted masks and examine them for segmentation accuracy. Look for regions of the image that are correctly segmented as cell membranes, as well as regions that are incorrectly segmented or missed completely.
   - **Compare the predicted masks with the original images:** By comparing the predicted masks with the original images, we can get a better sense of how accurate the segmentation is. Look for regions of the image where the predicted mask aligns well with the cell membrane boundaries in the original image, as well as regions where there are discrepancies.
   - **Qualitative analysis:** In addition to visual inspection, we can also perform qualitative analysis on the predicted masks. For example, we can compute the area of the segmented regions and compare it with the expected area based on prior knowledge of the cell membrane structure. We can also compare the predicted masks with known characteristics of cell membranes, such as their thickness or texture.
   - **Seek expert feedback:** Finally, we can seek feedback from domain experts, such as biologists or medical professionals, to evaluate the performance of our model. They can provide valuable insights into the accuracy of the segmentation and help we identify areas for improvement.
   - While qualitative evaluation is not as precise as quantitative evaluation using metrics like IoU or Dice coefficient, it can still provide valuable insights into the performance of our model. By combining visual inspection, qualitative analysis, and expert feedback, we can gain a better understanding of the strengths and limitations of our model and identify areas for improvement. It's also important to note that if we have access to a small subset of annotated data, we can use it to compute quantitative evaluation metrics and use those to guide our qualitative analysis.



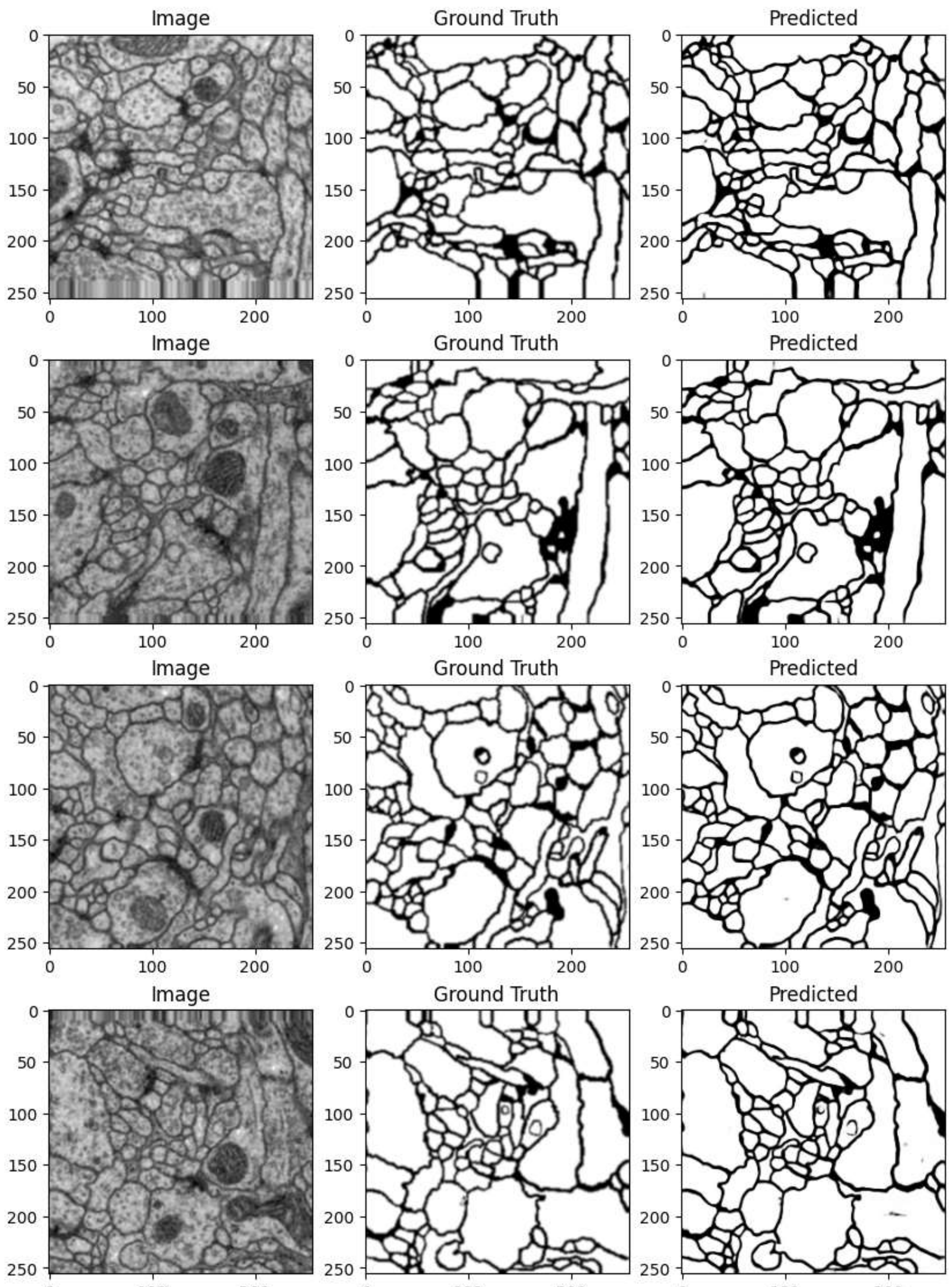


Image
Ground Truth
Predicted
0
50
100
150
200
250
0
100
200

5. **Here's some guidance on how to perform model analysis, model confidence analysis, check model calibration, and analyze feature importance (Some potential improvements):**
    - **Model Analysis:** Model analysis involves evaluating the performance of our model on various metrics, including accuracy, precision, recall, F1 score, and AUC-ROC. We can use the sklearn.metrics package to compute these metrics. We can also visualize the performance of our model using precision-recall and ROC curves. These curves plot the precision and recall, or true positive rate and false positive rate, of our model at different probability thresholds. A good model will have a high AUC-ROC score and a high precision-recall score.
    - **Model Confidence Analysis:** Model confidence analysis involves assessing the confidence of our model's predictions. We can compute the confidence of our model's predictions using the probability scores output by the model. One way to assess the confidence of our model's predictions is to compute the entropy of the predicted probabilities. High entropy indicates low confidence, while low entropy indicates high confidence.
    - **Check Model Calibration:** Model calibration refers to the alignment of the predicted probabilities with the true probabilities. A well-calibrated model will output probability estimates that are close to the true probabilities. We can assess the calibration of our model using calibration plots or reliability diagrams. These plots compare the predicted probabilities with the true probabilities for different probability bins. A well-calibrated model will have points close to the diagonal line.
    - **Analyze Feature Importance:** Analyzing feature importance involves identifying the most important features used by our model to make predictions. This can help we identify potential data leaks or overfitting issues, and also enhance the interpretability of our model.