0.1 ML Workflow

- Data collection
- Data cleaning, splitting
- Data exploration (EDA)
- Preprocessing, Feature engineering
- Feature selection, Model building
- Evaluation, model selection
- Test data predictions, visualizations, possible deployment

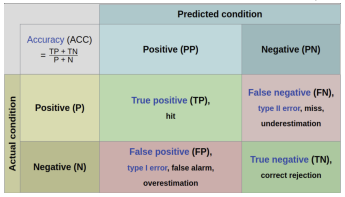
0.2 Confusion matrix

Instead of just reporting the accuracy, which tells us what proportion of the model predictions are correct, we can look into how the model predictions are correct and incorrect. A common way of doing this is to create a confusion matrix which shows how each class' actual and predicted label.

```
cm = ConfusionMatrixDisplay.from_estimator(pipe, X_valid, y_valid,
      values_format="d")
confusion_matrix(y_valid, pipe.predict(X_valid))
confusion_matrix(y_train, cross_val_predict(pipe, X_train, y_train))
```

- Perfect prediction has all values down the diagonal
- Off diagonal entries can often tell us about what is being mis-predicted
- Probably the missed fradulent transactions since they would be very costly! There is still some cost associated with investigating the non-fradulent transactions so we need to consider that as well, but to a lesser extent.

Dummy Confusion Matrix / Confusion Matrix



Disease test: It's worse to miss someone who actually has the disease (false negative) because they won't get the treatment they need. If we wrongly think someone has the disease (false positive), a doctor will check before treating them, so it's less harmful.

Spam filter: It's worse to mark a real email as spam (false positive) because the person might never see an important message. If a spam email gets through (false negative), you can just delete it yourself.

0.3 Confusion matrix with cross-validation

This adds up all the values of the TP, TN, FP, FN from the different CV folds, so that each part of the data is evaluated once (since we do crossvalidation with non-overlapping folds).

```
confusion_matrix(y_train, cross_val_predict(pipe, X_train, y_train))
ConfusionMatrixDisplay.from_predictions #used from CV same as above
```

0.4 Precision

Precision is the ability of the classifier to putting a positive label on a positive observation. You can remember this as "How precise are the model's predictions?".

$Precision = TP/(TP + FP)$

0.5 Recall

Recall is the ability of the classifier to find all the positive samples. You can remember it as "What proportion of the real positives the did the model recall.

$Recall = TP/(TP + FN)$

```
TP, FN, FP, TN = confusion_matrix(y_valid,
      pipe.predict(X_valid)).flatten()
precision_score(y_valid, pipe.predict(X_valid), pos_label='Fraud')
recall_score(y_valid, pipe.predict(X_valid), pos_label='Fraud')
```

0.6 Drawbacks of precision and recall - F1 Score

F1-score combines precision and recall to give one score, which could be used in hyperparameter optimization. It is the harmonic mean of precision and recall. In the F1 score, both the precision and recall are regarded as equally important.

```
f1_score = (2 * precision * recall) / (precision + recall)
f1_score(y_valid, pipe.predict(X_valid), pos_label='Fraud') #
      Recall and precision equally important
```

0.7 Fβ Score

However, in real life applications we might care more about one than the other but still want to hav a single number. For this we often use the general **Fβ Score**, where beta is the weight of recall vs precision.

```
fbeta_score(y_valid, pipe.predict(X_valid), pos_label='Fraud',
      beta=1) # Same as F1
fbeta_score(y_valid, pipe.predict(X_valid), pos_label='Fraud',
      beta=2) # Recall twice as important
```

0.8 Classification report

There is a convenient function called classification_report in sklearn which prints out the three metrics we have discussed all at once.

```
classification_report(y_valid, pipe.predict(X_valid))
```

0.9 Cross validation with different metrics

We can pass different evaluation metrics with scoring argument of cross_validate.

0.10 Precision/Recall tradeoff

By default, predictions use the threshold of 0.5. If predict_proba ¿ 0.5, predict "fraud" else predict "non-fraud". We can see this by manually typing in the 0.5 threshold and getting the same results as above.

Suppose for your business it is more costly to miss fraudulent transactions and suppose you want to achieve a recall of at least 75% for the "fraud" class. (Also called **Operating point**)

If you identify more things as "fraud", recall is going to increase but there are likely to be more false positives.

Decreasing the threshold

- Using a lower bar to predict fraud.
- You accept more false positives in exchange for gaining more true positives.
- Recall will either stay the same or increase.
- Precision will likely decrease, though it can sometimes increase if all newly added predictions are true positives.

Increasing the threshold

- Using a higher bar to predict fraud.
- Recall will decrease or stay the same.
- Precision will likely increase, though it may decrease if true positives drop without reducing false positives.

0.11 Precision-recall curve

Often, when developing a model, it's not always clear what the operating point will be and to understand the the model better, it's informative to look at all possible thresholds and corresponding trade-offs of precision and recall in a plot. This type of plot is often called a PR curve, and the top-right would be a perfect classifier (precision = recall = 1).

- Usually the goal is to keep recall high as precision goes up.

0.12 AP score

- one number summarizing the PR plot
- One way to do this is by computing the area under the PR curve.
- Called average precision
- AP score has a value between 0 (worst) and 1 (best).

0.13 AP vs. F1-score

- F1 score is for a given threshold and measures the quality of predict.
- AP score is a summary across thresholds and measures the quality of predict_proba
- Optimizing towards a high F1 score means that you find the model that performs the best at the default decision threshold while considering both recall and precision
- Optimizing towards a high AP score means that you get the model that performs the best over all possible decision thresholds, which could give you more flexibility in your tradeoffs between precision and recall when deciding on a decision threshold.

0.14 Receiver Operating Characteristic (ROC) curve

- Another commonly used tool to analyze the behavior of classifiers at different thresholds.
- Similar to PR curve, it considers all possible thresholds for a given classifier given by predict_proba but instead of precision and recall it plots false positive rate (FPR) and true positive rate (TPR or recall).
- FPR = FP/(FP+TN)
- The ideal curve is close to the top left. Classifier with high recall while keeping low false positive rate.

0.15 Which type of error is important in imbalanced

- False positives (FPs) and false negatives (FNs) have quite different real-world consequences.
- In PR curve and ROC curve, we saw how changing the prediction threshold can change FPs and FNs.
- We can then pick the threshold that's appropriate for our problem.
- if we want high recall, we may use a lower threshold (e.g., a threshold of 0.1). We'll then catch more fraudulent transactions. Let's first see the report with the standard 0.5 threshold.

0.16 Handling imbalanced

Undersampling, Oversampling, Random oversampling, SMOTE: Synthetic Minority Over-sampling Technique, Changing the training procedure - class_weight. All sklearn classifiers have a parameter called class_weight, For example, maybe a false negative is 10x more problematic than a false positive.

- Recall is much better but precision has dropped a lot; we have many false positives.
- We can optimize class_weight using hyperparameter optimization for your specific problem.
- Changing the class weight will generally reduce accuracy.
- if we want high recall, we may use a lower threshold (e.g., a threshold of 0.1). We'll then catch more fraudulent transactions. Let's first see the report with the standard 0.5 threshold.

0.17 Stratified Splits

- Well, it's no longer a random sample, which is probably theoretically bad, but not that big of a deal.
- It can be especially useful in multi-class, say if you have one class with very few cases.