

1 ML Workflow

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

2 Confusion matrix

```
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

Actual condition	Predicted condition	
	Positive (PP)	Negative (PN)
	Accuracy (ACC) $= \frac{TP+TN}{P+N}$	
Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

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.

2.1 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
```

2.2 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?".

$Precision = TP / (TP + FP)$

2.3 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 they 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')
```

2.4 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
```

2.5 Fβ Score

However, in real life applications we might care more about one than the other but still want to have 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
```

2.6 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))
```

3 Cross validation with different metrics

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

3.1 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.

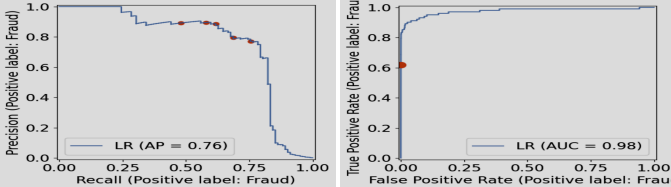
3.2 Precision-recall curve - Fig1

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.

3.3 Receiver Operating Characteristic (ROC) curve - Fig2

- 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.



3.4 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).

3.5 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.

4 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 thresh-

old of 0.1). We'll then catch more fraudulent transactions.

4.1 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.

4.2 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.

4.3 Tuning alpha hyperparameter of Ridge

- This is like C in LogisticRegression but, annoyingly, alpha is the inverse of C.
- Smaller alpha: lower training error (overfitting)
- Higher values of alpha means a more restricted model.
- The values of coefficients are likely to be smaller for higher values of alpha compared to lower values of alpha.
- RidgeCV, which automatically tunes alpha based on cross-validation

5 Regression scoring functions

In sklearn for regression problems, using r2_score() and .score() (with default values) will produce the same results

5.1 Mean squared error (MSE)

$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$ - Affected by outliers - always non-negative - lesser better - hence we take negative value for CV

5.2 Root mean squared error (RMSE)

$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$ - always non-negative - lesser better - hence we take negative value for CV

5.3 Mean absolute error (MAE) - Less sensitive to outliers

$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$

5.4 Mean absolute percentage error (MAPE) - frames the error in terms of a proportion/percentage instead of an absolute number.

$MAPE = \frac{100}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|$

5.5 R² - compare our model's errors to the errors in a baseline model which always predicts the mean.

$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$

5.6 Different scoring functions with cross-validate

The default in sklearn's CV is to try to make the scoring metric as big as possible, which makes sense for classification metrics such as accuracy, precision, recall, etc (and R2). However, for many our regression metrics, smaller is better. Because of this, the version used for CV needs to be inverted to be negative, which is reflected in the name of these CV scoring methods, such as neg_mean_squared_error, and neg_root_mean_squared_error.

5.7 Transforming the targets

When you have prices or count data, the target values can often be skewed. A common trick in such cases is applying a log transform on the target column to make it more normal and less skewed. Linear regression will usually have better performance on data that is more normal.

6 Feature Engineering

Good features would ideally: Capture most important aspects of the problem Allow learning with fewer examples Generalize to new scenarios. There is a trade-off between simple and expressive features: With simple features overfitting risk is low, but scores might be low. With complicated features scores can be high, but so is overfitting risk.

6.1 Examples

counting-based methods like decision trees - discretization
For distance-based methods like KNN - standardization
For regression-based methods like linear regression - polynomial