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

```
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 mispredicted

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

```
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.

$Precision = TP/(TP + FP)$

2.3 Recall

Recall is the ability of the classifier to find all the positive samples.

$Recall = TP/(TP + FN)$

```
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 works for hyperparameter opt. 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
classification_report(y_valid, pipe.predict(X_valid)) # get all
     values like recall precision
```

2.5 Fβ Score

When we care about recall more than percision and vice versa, use the general **Fβ Score**, where beta is the weight of recall vs precision. beta = 1 is f1 score.

2.6 Precision/Recall tradeoff

By default, predictions use the threshold of 0.5. If predict_proba $\geq$ 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.

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.

2.7 Precision-recall curve - Fig1

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

2.8 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.

2.9 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).

2.10 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.

3 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.

3.1 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 Regression scoring functions

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

4.1 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

4.2 Root mean squared error (RMSE)

RMSE = $\sqrt{MSE}$ - always non-negative - lesser better - hence we take negative value for CV

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

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

4.4 Mean absolute percentage error (MAPE)

MAPE = $\frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$- frames the error in terms of a proportion/percentage instead of an absolute number.

4.5 R^2 - compare our model's errors to the errors in a baseline model which always predicts the mean.

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

4.6 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.

5 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.

5.1 Polynomial

Linear models learn lines, planes, or hyperplanes. Poly can learn, hyperbolas, quadratic function. Increasing deg makes the model more complex, overfitting, pick the degree of polynomial using hyperparameter optimization. Suitable for regression problems, they can also help in classification. 2-features - Planes — Circles - Poly with 2 degree — Polynomial feature engineering first then Standard scaler to avoid the polynomial features from having much larger absolute magnitude than the original features, which would be an issue for any distance based calculations and for interpretation of coefficients.

5.2 Feature discretization / binning

Transforming numeric features into categorical features is called bucketing or binning.

5.3 Bag-of-words model - Text - N-grams

Transforming numeric features into categorical features is called bucketing or binning.

6 Feature Selection

- Correlation among features might make coefficients completely uninterpretable.
- Fairly straightforward to interpret coefficients of ordinal features.
- In categorical features, helpful to consider one category as a reference point and think about relative importance.
- For numeric features, relative importance is meaningful after scaling.
- You have to be careful about the scale of the feature when interpreting the coefficients. (polynomial)
- Explaining the model is not same as explaining the data
- Coefficients tell us only about the model and they might not accurately reflect the data

6.1 Model-based feature selection

Keep only the most important features. We can put the feature selection transformer in a pipeline. preprocessor - select_lr - randomForestRegressor = select_lr selects important features before running model

```
pipe_rf_model_based['randomforestregressor'].n_features_in_ #gives
     features count
```

6.2 Recursive feature elimination (RFE)

But it's different in the sense that it iteratively eliminates unimportant features. It builds a series of models. At each iteration, discards the least important feature according to the model. Computationally expensive.

6.3 RFE with cross-validation (RFECV)

We arbitrarily picked 120 features before. Use RFECV which uses cross-validation to select number of features. Slow because there is cross validation within cross validation

6.4 Forward or backward selection

Shrink or grow feature set by removing or adding one feature at a time. Makes the decision based on whether adding/removing the feature improves the CV score or not. Very slow. selects d/2 features

6.5 Warnings about feature selection

A feature's relevance is only defined in the context of other features. If features can be predicted from other features, you cannot know which one to pick. Relevance for features does not have a causal relationship.