1. Alpha in Ridge is the lower the more complex the model is, when alpha = 0, same as linear regression.

2. For Confusion matrix, the y axis is actual, x axis is predicted.

3. PR curve (precision-recall curve), x axis is precision and y is recall

4. False positive rate = FP / FP+TN (among all negatives, how many negatives we mislabel)

5. ROC curve which is **FPR-x vs TPR-y**: it considers all possible thresholds for a given classifier given by predict\_proba, which is to analyze the behavior of classifiers at different thresholds.

y axis = true positive rate (recall), x axis = false positive rate

(**AUC of 0.5 means random chance, AUC of 1.0 means all positive points have a higher score than all negative points**.)

6. MAPE = abs(mean((pred\_train - y\_train) / y\_train) \* 100.0)

almost every single lecture after midterm, please review them one by one!

**Lecture 11** about random forest (the questions asked in lectures)

**How Random forest works:**

**1. Create a collection (ensemble) of trees. Grow each tree on an independent bootstrap sample from the data.**

**2. At each node:**

**2.1 Randomly select a subset of features out of all features (independently for each \*\*node\*\*).**

**2.2 Find the best split on the selected features.**

**2.3 Grow the trees to maximum depth.**

**3.Prediction time**

**Vote the trees to get predictions for new example.**

2. Stacking: Use their outputs as inputs to another model.

* By default for classification, it uses logistic regression.
  + We don’t need a complex model here necessarily, more of a weighted average.
  + The features going into the logistic regression are the classifier outputs, not the original features!
  + So the number of coefficients = the number of base estimators!

Graphical user interface, text, application, email

Description automatically generated

What’s going on in here?

* It is **doing cross-validation by itself by default** (see [documentation](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html))
  + It is **fitting the base estimators on the training fold**
  + And the **predicting on the validation fold**
  + And then **fitting the meta-estimator (final estimator** – logistics regression by default) on that output (**on the** **validation fold**)

**Lecture 12** – feature importance - Eli5 and SHAP for feature importances

~ eli5 can be used to get feature importances for **non sklearn** models.

~ SHAP

**Start at a base rate** (e.g., how often people get their loans rejected).

**Add one feature at a time and see how it impacts the decision**.

Also force plot, might take a look again

**Lecture 13** about feature selection (search and score - forward and backward selection)

1. RFE (Recursive feature elimination)

~ Build a series of models. At each iteration, discard the least important feature according to the model.

~ computationally expensive (slow)

Basic idea

1. fit model

2. find least important feature

3. remove

4. iterate.

**Detailed steps:**

**1. Decide k, the number of features to select.**

**2. Assign importances to features, e.g. by fitting a model and looking at coef\_ or feature\_importances\_.**

**3. Remove the least important feature.**

**4. Repeat steps 2-3 until only k features are remaining.**

How do we know what value to pass to n\_features\_to\_select?

Use RFECV which uses cross-validation to select number of features.

**2. Search and score**

Example: Suppose you have three features: A,B,C

Compute score for S={}

Compute score for S={A}

Compute score for S={B}

Compute score for S={C}

Compute score for S={A,B}

Compute score for S={A,C}

Compute score for S={B,C}

Compute score for S={A,B,C}

Return S with the best score.

How many distinct combinations we have to try out?

**3. 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**

Text, letter

Description automatically generated

Difference between this forward/backward selection and RFE is the former’s decision based on CV score and the latter based on coef\_or feature\_importance

**Lecture 14** about k-means clustering (how to choose k=number of clusters? -> elbow & sihouette method)

**1. The Elbow method**

**~ Looks at the squared sum of intra-cluster distances (a.k.a inertia)**

~ The inertia decreases as K increases but the problem is that we can't just look for a k that minimizes inertia (If each point is a cluster, the inertia is 0!!)

~ We evaluate the tradeoff "small k" vs "small intra-cluster distances"

Elbow method describes the tradeoff between them and plot a graph

**2. The silhouette method**

**~ Not dependent on the notion of cluster centers**

**~ calculated using the mean intra-cluster distance (a) and mean nearest-cluster distance (b) for each sample**

Chart

Description automatically generated

How to interpret the **silhouette** graphs!!!!! (Important)

~ Higher values indicate well-separated clusters. **Unlike inertia, larger values are better because they indicate that the point is further away from neighbouring clusters.**

~ The **thickness** of each silhouette indicates **the cluster size**.

~ Unlike inertia, the overall silhouette score gets worse as you add more clusters because you end up being closer to neighbouring clusters.

**~ Thus, as with inertia, you will not see a “peak” value of this metric that indicates the best number of clusters.**

~ The shape of each silhouette indicates the “goodness” for points in each cluster.

~ The length (or area) of each silhouette indicates the goodness of each cluster.

~ A slower dropoff (**more rectangular) indicates more points are “happy” in their cluster.**

**~ It always converges. Convergence is dependent upon the initial centers and it may converge to a sub-optimal solution.**

**Lecture 15 – DBScan**

~ we CANNOT use **elbow method** to examine the goodness of clusters created with DBSCAN

~ but **silhouette** method could

* **Unlike K-Means, there is no predict method.**
  + **DBSCAN only really clusters the points you have, not “new” or “test” points.**
* **Pros**
  + **Can learn arbitrary cluster shapes**
  + **Can detect outliers**
* **Cons**
  + **Cannot predict on new examples.**
  + **Needs tuning of two non-obvious hyperparameters (eps and min\_samples)**
  + **DBSCAN doesn’t do well when we have clusters with different densities.**

**Lecture 16 - NLP**

Sparse representation: term-term co-occurrence matrix

Dense representation: Word2Vec (Based on **distributional hypothesis** and **vector space model**)

Topic modelling:

Input:

1. A large collection of documents
2. A value for the hyperparameter K

Output:

1. For each topic what words describe that topic
2. For each document, what topics are expressed by the document.

16.2 Text preprocessing

### Tokenization

* Sentence segmentation
  + Split text into sentences
* Word tokenization
  + Split sentences into words

Types and tokens

~ Type: an element in the vocabulary

~ Token: an instance of that type in running text

### Punctuation and stopword removal

* The most frequently occurring words in English are not very useful in many NLP tasks.
  + Example: the , is , a , and punctuation
* Probably not very informative in many tasks

**Lemmatization: want to ignore some morphological differences between works**. E.g if your search term is "studying for ML quiz" you might want to include pages containing "tips to study for an ML quiz" or "here is how I studies for my ML quiz".

It converts inflected forms into the base form.

**Stemming: has a similar purpose (reduce words by chopping them)**

Say "automates", "automatic" could be reduced to "automat" which might not be an english word

**Lecture 17 – Multiclass classification and computer vision**

Two kind of “hacky” ways to reduce multi-class classification into binary classification

### One vs. Rest

* 1v{2,3}, 2v{1,3}, 3v{1,2}
* Learn a binary model for each class which tries to separate that class from all of the other classes.
* If you have k classes, it’ll train k binary classifiers, one for each class. (This makes our interpretation harder)
* **Trained on imbalanced datasets containing all examples. (Every time you try to the train on a binary classification, the rest of the classes become one majority and your own class becomes the minority)**
* Given a test point, get scores from all binary classifiers (e.g., raw scores for logistic regression).
* The classifier which has the highest score for this class “wins” and that’s going to be the prediction for this class.
* Since we have one binary classifier per class, for each class, we have coefficients per feature and an intercept.

### One Vs. One approach (most likely works a little bit better)

* Build a binary model for each pair of classes.
* 1v2, 1v3, 2v3
* Trains n×(n−1)/2 binary classifiers **(more time-consuming)**
* Trained on **relatively balanced** subsets (pro against one vs rest)

Transfer learning

* Instead, a common practice is to download a pre-trained model and fine tune it for your task.
* This is called **transfer learning**.

Lecture 18 – time series (skip)

**Lecture 19 – Survival analysis**

Frequently, there will be some kind of **censoring** which will not allow you to observe the exact time that the event happened for all units/individuals that are being studied. In such a case, we cannot use linear regression

Graphical user interface, application

Description automatically generated

## **Cox proportional hazards model**

* The Cox proportional hazards model is a commonly used model that allows us to interpret how features influence a censored tenure/duration.
* You can think of it like linear regression for survival analysis: we will get a coefficient for each feature that tells us how it influences survival.
* The proportional hazard model works multiplicatively, like linear regression with log-transformed targets.
* Looks like month-to-month leads to more churn, two-year contract leads to less churn; this makes sense!!!

Table

Description automatically generated

Lecture 21 – communication

Principles of good explanations

1. Concepts then labels, not the other way around

2. Bottom-up explanations

3. New ideas in small chunks

4. Reuse your running examples

5. Approach from all angles

6. When experimenting, show the results asap

7. Interesting to you != useful to the reader (aka it’s not about you)

Past Final exam

Q1. should use ordinal, quite make sense.

Interesting questions on Piazza

* 1. <https://piazza.com/class/ky0j51i4ud64t5?cid=318> (About recall, precision, macro average)
  2. <https://piazza.com/class/ky0j51i4ud64t5?cid=310> (About feature selection, especially Q2)
  3. <https://piazza.com/class/ky0j51i4ud64t5?cid=303_f1> (About survival analysis)
     + The Cox proportional hazards model (cph above) assumes the effect of a feature is the same for all customers and over all time. (**False**, The actual assumption is that the *ratio* of the hazards for any two individuals is constant over time. We did not go into this in detail as it is a bit difficult to evaluate.)
  4. <https://piazza.com/class/ky0j51i4ud64t5?cid=359> (RFE – lower ranks being removed first)