# 8 Advanced Practice

## Handling Imbalanced Datasets

An _________________ *dataset* has one class under-represented.

e.g. Fraudulent e-commerce transactions are much less common than genuine ones. Noise puts genuine ones on the wrong side of the desired decision boundary, moving it to a ___________ place.

Possible solutions:

- For SVM, we can assign __________________________ to the minority class.

  e.g. For binary SVM, instead of finding soft-margin's $\text{argmin}_{\mathbf{w},b}\left[\frac{1}{2}||\mathbf{w}||^2 + C\frac{1}{N}\sum_{i=1}^{N}\max(0, 1 - y_i(\mathbf{w}\mathbf{x}_i + b))\right]$, we find something like

  $$\text{argmin}_{\mathbf{w},b}\left[\frac{1}{2}||\mathbf{w}||^2 + C\frac{1}{N}\left(C_1 \sum_{\{(\mathbf{x}_i,y_i)|y_i=-1\}} \max(0, 1 - y_i(\mathbf{w}\mathbf{x}_i + b)) + C_2 \sum_{\{(\mathbf{x}_i,y_i)|y_i=+1\}} \max(0, 1 - y_i(\mathbf{w}\mathbf{x}_i + b))\right)\right]$$

  where $C_1$ and $C_2$ are regularization parameters that can be set as _____________ .

  e.g. See Burkov's Figure 8.1 on p. 98 (p. 3 of `www.dropbox.com/s/im1s2skkaikzrrs/Chapter8.pdf?dl=0`).

  The same problem (before re-weighting imbalanced data) arises with most algorithms.

- ____________________ adds multiple copies of minority class examples.
- ____________________ randomly removes some majority class examples.
- Create _____________ examples by combining randomly sampled feature values from several examples of minority class.

Do `train_test_split()` _____________ addressing imbalance so that test data are ________________.

## Python

- The `svm.SVC()` we know[1] has a `class_weight` parameter:
  - The default `None` gives weights $C_1 = C_2 = 1$ to each class.
  - It can be a dictionary of `label:value` pairs (where `value` $> 0$) like `{0: C_1, 1: C_2}`.
  - Using `'balanced'` gives weights inversely proportional to class counts in training data as `N / (n_classes * np.bincount(y))`; e.g.

[1] `svm.SVC(kernel='linear', C=1)` for soft-margin linear SVM (or `C=1000` for hard-margin), `svm.SVC(kernel='rbf', C=1, gamma='scale')` for kernel trick for nonlinear boundary

```
y = np.array([0, 0, 0, 0, 0, 1]) # 5 zeros, 1 one
N = y.shape # 6
counts = np.bincount(y) # array([5, 1])
n_classes = counts.shape # 2
C_1, C_2 = N / (n_classes * counts) # 0.6, 3
```

- For over- and undersampling,[2]
  - ```
    from imblearn.over_sampling import RandomOverSampler
    rs = RandomOverSampler(random_state=None)
    X_resampled, y_resampled = rs.fit_resample(X, y)
    ```
  - ```
    from imblearn.under_sampling import RandomUnderSampler
    rs = RandomUnderSampler(random_state=None)
    X_resampled, y_resampled = rs.fit_resample(X, y)
    ```

  e.g.

```
X = np.array([1, 2, 3, 4, 5, 6]).reshape(-1, 1)
y = np.array([0, 0, 0, 0, 1, 1])

rs = RandomOverSampler()
X_resampled, y_resampled = rs.fit_resample(X, y)
print(f'Oversampling: X_resampled={X_resampled},\ny_resampled={y_resampled}')

rs = RandomUnderSampler()
X_resampled, y_resampled = rs.fit_resample(X, y)
print(f'Undersampling: X_resampled={X_resampled},\ny_resampled={y_resampled}')
```

To learn more:

- User guide:

  `https://scikit-learn.org/stable/modules/svm.html#unbalanced-problems`

  `https://imbalanced-learn.org/stable/over_sampling.html`

  `https://imbalanced-learn.org/stable/under_sampling.html`

- Reference manual:

  `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

  `https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html`

  `https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html`

- Example:[3]

  `https://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane_unbalanced.html`

---

[2] Do "New > Terminal" and then run `conda install -c conda-forge imbalanced-learn` to install package.

[3] Click on "launch binder" to run it online. Note `class_weight={1: 10}`, which leaves class 0 at the default weight of 1). Also try `class_weight={0: 1, 1: 1}` (balanced) and `class_weight={0: 12, 1: 10}` (almost balanced).

## Combining Models

While ensemble methods like random forests combine several similar weak models, we can also combine different ____________ models:

- ____________ the predictions (regression) or scores (classification) of several models.
- *Majority vote* applies several models and returns the __________________ predicted class. (Resolve a tie by choosing randomly or returning an error (or use an odd number of models).)
- ____________ builds a meta-model whose input is the output of several base models. e.g. To combine models $f_1$ and $f_2$ that predict from the same set of classes, create a training example $(\mathbf{x}_i', y_i')$ for the stacked model as $(\mathbf{x}_i' = [f_1(\mathbf{x}_i), f_2(\mathbf{x}_i)], y_i' = y_i)$[4] and train a meta-model on the new examples. Tune hyperparameters with cross-validation. Comparatave notes:
  - Stacking uses ______________________ from the base models (scores across $C$ class labels) than averaging or majority voting (single best class label from among $C$ labels).
  - Stacking uses ____________ models on the same data, while bagging uses the ____________ model on different (bootstrap resampled) data.
  - Stacking uses ________________ to combine predictions from base models, while boosting uses a sequence of models in which the next model tries to correct the current one.

Base models should be ____________ by being made from different features or different algorithms.

### Python

- `from sklearn.ensemble import StackingClassifier, StackingRegressor`

  `clf = StackingClassifier(estimators, final_estimator=None)`

  `model = StackingRegressor(estimators, final_estimator=None)`

  `estimators` is a list of tuples `(string name, estimator)` giving the models to be stacked.

  `final_estimator` uses the output of `estimators` as input.

To learn more:

- User guide: `https://scikit-learn.org/stable/modules/ensemble.html#stacking`
- Reference manual:

  `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html`

  `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingRegressor.html`
- Example:

  `https://scikit-learn.org/stable/auto_examples/ensemble/plot_stack_predictors.html`

---

[4] $f_1(\mathbf{x}_i)$ is the output of `clf1.predict_proba(x)` or `clf1.decision_function(x)` or `model1.predict(x)`.

**Algorithm Efficiency**

______________ *of algorithms* reveals the computational complexity of algorithms in terms of the time (or memory or other resources) they require. We use ______________ *notation* to write time as a function of input size $N$, and then ______________ constants and lower-order terms.

- Suppose a program running on input of size $n$ has run time $f(n)$ seconds.
- Big-$O$ gives an upper bound on run-time to within a constant factor. A function $f(n)$ is said to be $O(g(n))$ if there exist constants $C$ and $N$ such that $f(n) < C \cdot g(n)$ for all ______________. (Draw picture.)
- Read "$f(n) = O(g(n))$" as "$f(n)$ is big-$O$ of g(n)."
- Here are some typical $g(n)$ functions in increasing order:
  - $g(n) = 1$, e.g. ____________________________ by index `i`
  - $g(n) = \log_2(n)$, e.g. ____________________________ in sorted array
  - $g(n) = n$, e.g. ____________________________
  - $g(n) = n \log_2(n)$, e.g. clever comparison ______________
  - $g(n) = n^2$, e.g. ____________________________
  - $g(n) = n^3$, e.g. matrix _____________________, $C = AB$ via $c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$
  - $g(n) = n!$, e.g. traveling salesman via _____________________
- Just reading a data set of size $n$ is $O($_______$)$, so an $O(n)$ algorithm (that runs only once) counts as ________ . Since $\log_2(n)$ is small for typical $n$, an $O(n \log_2(n))$ algorithm is often fast enough. Programs taking $O(n^2)$ or more time may work for small $n$ but can be _______________ for large $n$.
- The ______________ of the algorithm usually matters a lot more than processor speed, coding skill, programming language, etc.
- If we cannot figure out the $O()$ formula, we can __________ the code for several dataset sizes $N$ and make a graph of time vs. $N$. e.g.

```
start = time.time() # get time in seconds since "time started" (often 1/1/1970)
# ... code that requires timing goes here ...
end = time.time()
seconds = end - start
print(f'The code took {seconds} seconds.')
```

- When the time is too long on $N$ examples, work with a ___________ randomly-selected subset.

To learn more:

- `https://scikit-learn.org/stable/computing/computational_performance.html`
- `https://scikit-learn.org/stable/developers/performance.html`
- `https://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms`

### Multicore computing to speed up computation

In *multicore* computing, an algorithm is run ___________________ on multiple CPU cores.[5]

#### Python

Some estimators support multicore computing via an _____________ parameter: set `n_jobs=None` to use one core, `n_jobs=n` to use `n`, or `n_jobs=-1` to use all. Find #CPUs via

```
import os # operating system interfaces (https://docs.python.org/3/library/os.html)
n_CPU = os.cpu_count()
```

Multicore methods include:

- §3: `KNeighborsClassifier()`, `KNeighborsRegressor()`
- §5: `cross_val_score()`, `GridSearchCV()`, `RandomizedSearchCV()`
- §5: `permutation_importance()`
- §7: `BaggingRegressor()`, `BaggingClassifier()`, `RandomForestRegressor()`, `RandomForestClassifier()`
- §8: `StackingClassifier()`, `StackingRegressor()`

To learn more:

- User guide: `https://scikit-learn.org/stable/computing/parallelism.html`

[5] Amdahl's law (`https://en.wikipedia.org/wiki/Amdahl%27s_law`) says "Don't expect ___________________."