DATA 301: Supervised Learning, splitting the dataset

Unsupervised Learning

Unsupervised

- Unlabeled Data
- Model does not know correct result during training
- Model finds hidden patterns in data
- Goal is to train the model to detect these patterns in unseen data
- Examples: DBSCAN, HDBSCAN, Mean Shift

	Gender	Annual Income (k$)	Birthday	spending_total	cluster
0	-1.134617	-1.749774	-1.424332	-0.430005	?
1	-1.134617	-1.749774	-1.281071	1.199919	?
2	0.881354	-1.711089	-1.352702	-1.710660	?
3	0.881354	-1.711089	-1.137810	1.044688	?
4	0.881354	-1.672403	-0.564766	-0.391197	?

Algorithm has to figure this out

Supervised

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Known Target

Supervised

- Labeled Data
- Model knows correct result during training
- Uses correct result to adjust model parameters during training
- Goal is to train model to pick correct result on unseen data
- Examples: Regressions, Decision Trees, Random Forest, Neural Networks

Supervised Data Setup (easy part)

Still need to do pre-processing (no standardization for tree based methods needed although it does not hurt)

Additionally:

- Need to divide data into 2 portions; train, test
- When we get to cross validation we need 3 portions; train, test, validation
- Typically 80% train 20% test (or 70-30, or 90-10 depends on amount of data you have)
- Calculate standardization metrics on the train set only. Apply these IDENTICAL metrics to the test set.
- If you violate this rule, test set information will leak over to the train set.

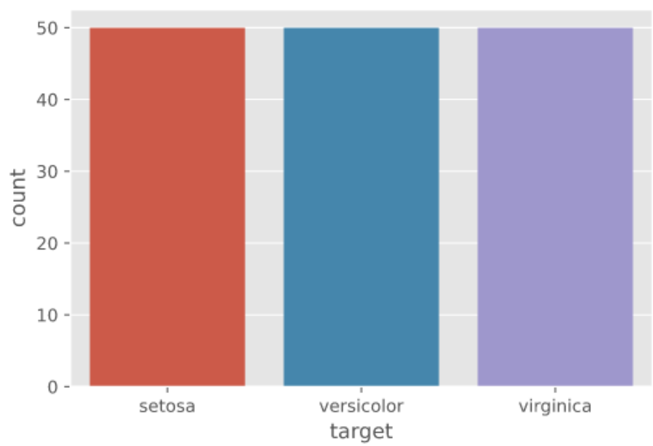
Train/Test split – balanced datasets

If your dataset has roughly the same number of labeled classes then its easy to do a train test split. Here we load the iris dataset, 77% of it is used to train, 33% to test.

```
from sklearn.model_selection import train_test_split
from sklearn import datasets
iris = datasets.load_iris()

X, y = iris.data, iris.target
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33,random_state=42)
```

Notice we have an equal number of iris types

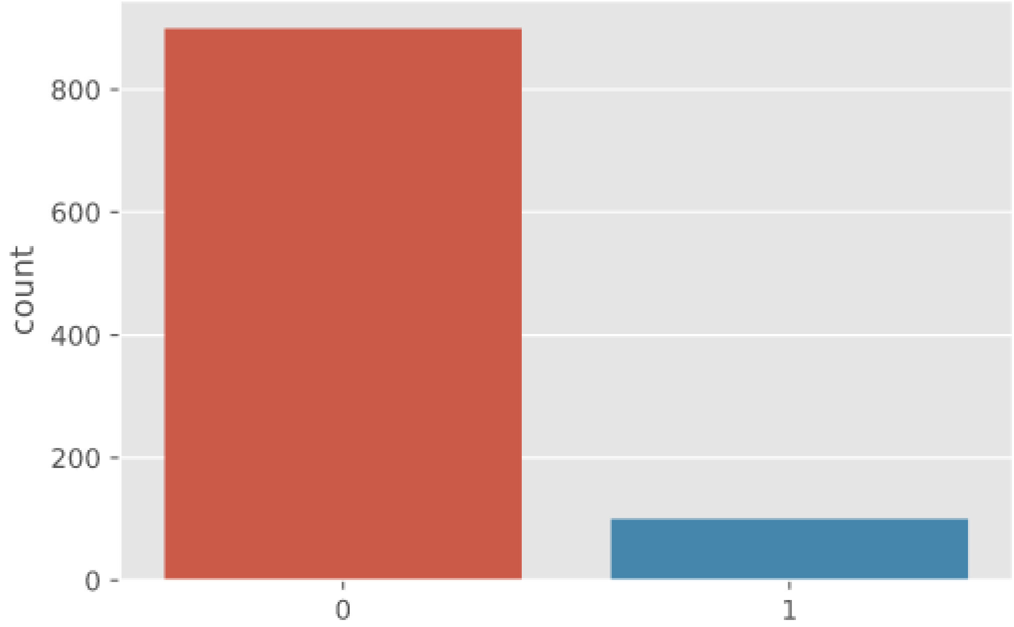
```
ax=sns.countplot(x="target", data=df)
```



Train/Test split – imbalanced datasets

Dataset has more of one class than others.
This is very common, think of cancer diagnosis

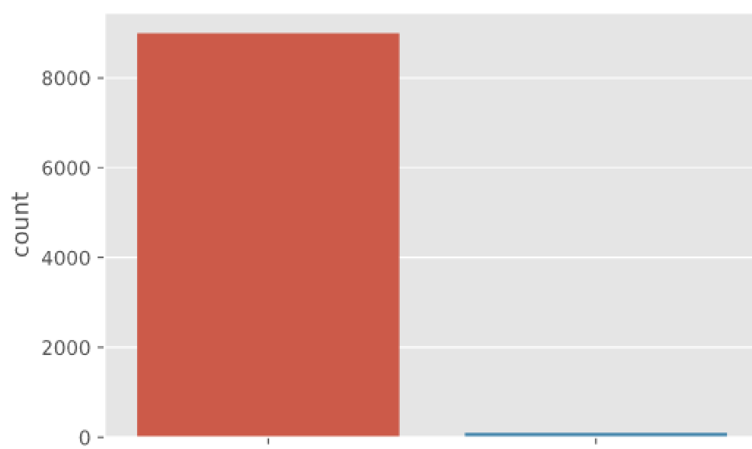
```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=[900,100], centers=None, cluster_std=[10.0, 2],random_state=22,n_features=2)
ax=sns.countplot(x=y)
```



Train/Test split – imbalanced datasets,

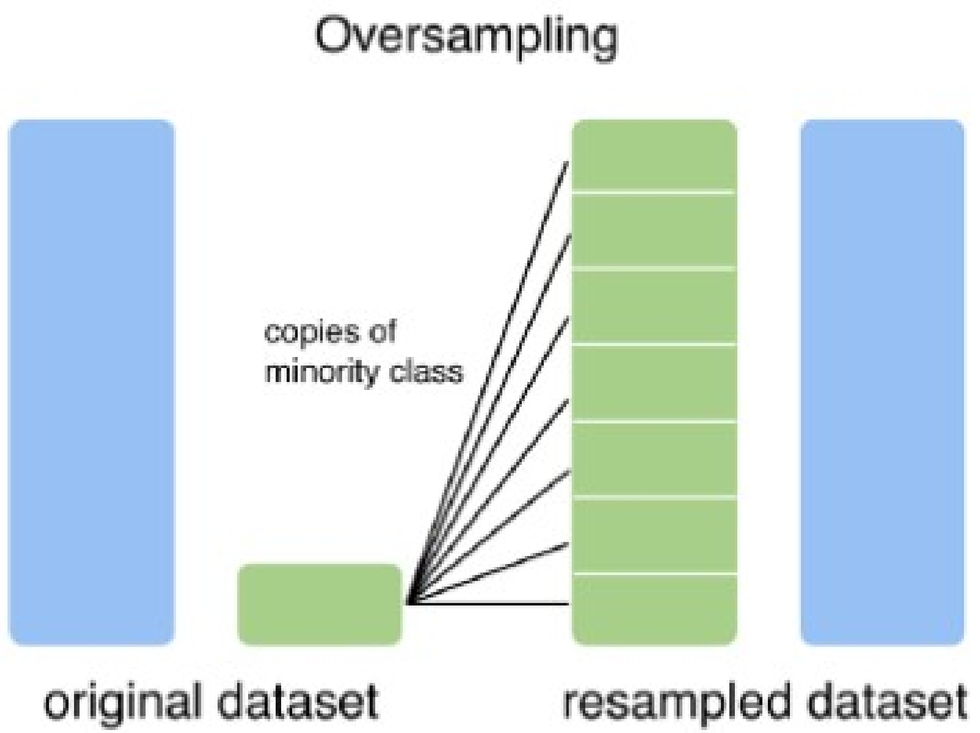
What if dataset is very imbalanced?
For fraud detection, 99.9% of transactions are legitimate.
Which means a fraud dataset consists of almost all legitimate transactions.
BTW for this case, if your evaluation metric is accuracy, a model may learn to always say its <u>not</u> fraud (99.9% accuracy!). See notebook on website for example.

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=[9000,100], centers=None, cluster_std=[10.0, 2],random_state=22,n_features=2)
ax=sns.countplot(x=y)
```



Train/Test split – imbalanced datasets, Guides

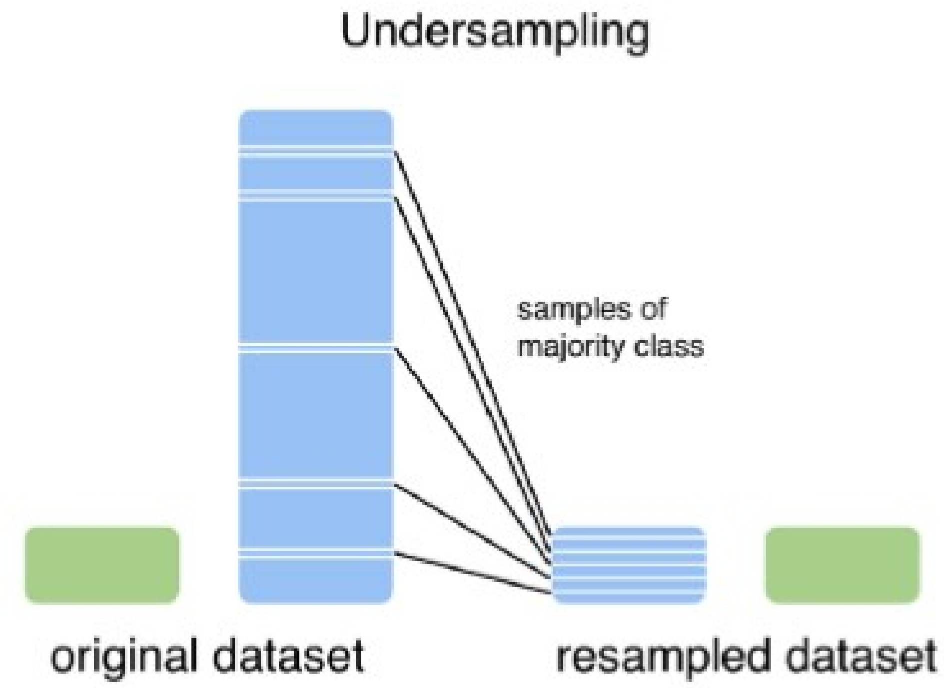
If you don’t have a lot of data (<10,000 samples)
oversample the minority



* https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
https://medium.com/strands-tech-corner/unbalanced-datasets-what-to-do-144e0552d9cd/

Train/Test split – imbalanced datasets, Guides

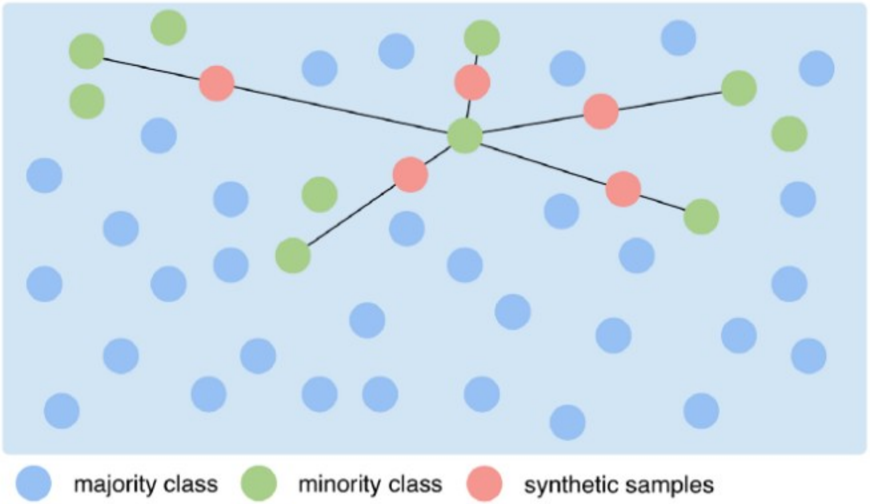
If you have a lot of data (>>10,000 samples)
undersample the majority



* https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
https://medium.com/strands-tech-corner/unbalanced-datasets-what-to-do-144e0552d9cd/

Train/Test split – imbalanced datasets, Guides

Try to generate synthetic samples. There SMOTE package uses kNN to generate synthetic points along lines connecting members of the minority class.



Because SMOTE selects a point between existing points, the boundary of the minority class will not grow

* https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
https://medium.com/strands-tech-corner/unbalanced-datasets-what-to-do-144e0552d9cd/

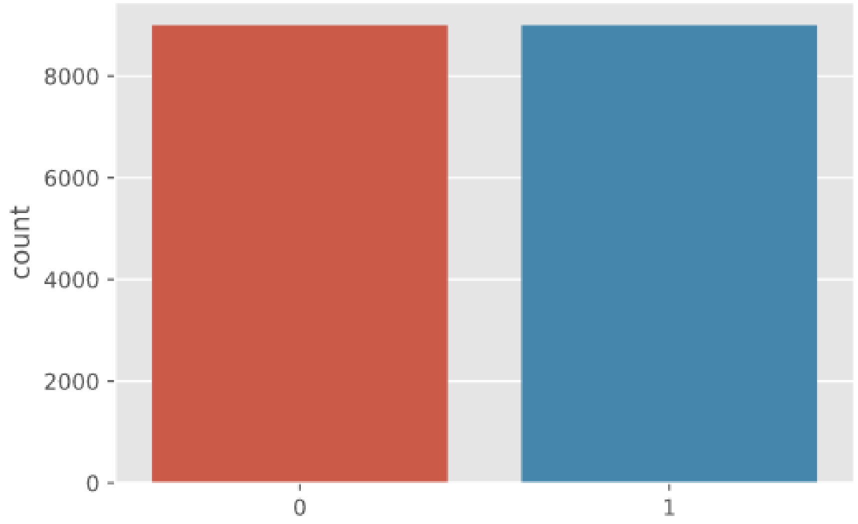
Train/Test split – imbalanced datasets, Guides

Try to generate synthetic samples. There SMOTE package uses kNN to generate synthetic points along lines connecting members of the minority class.

```
# !conda install -c conda-forge imbalanced-learn -y
```

•••

```
import imblearn
oversample = SMOTE()
X_new, y_new = oversample.fit_resample(X, y)

ax=sns.countplot(x=y_new)
```



* https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
https://medium.com/strands-tech-corner/unbalanced-datasets-what-to-do-144e0552d9cd/

Train/Test split – imbalanced datasets, Guides

Maybe the easiest (or hardest) of all;
Collect more data from the minority class

* https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
https://medium.com/strands-tech-corner/unbalanced-datasets-what-to-do-144e0552d9cd/

Train/Test split – temporal or time series data

Time series data such as:

- Daily or Weekly sales
- Stock closing prices
- Daily temperature fluctuations

Temporal data has a lot of complexity, there is generally an underlying seasonality and a trend.

You can not randomly sample train/test splits with temporal data. The test set must come from the latest data, or you risk information from later data leaking into earlier data.

We will (hopefully) address temporal data later

Train/Test split – prefer stratified splits for classification

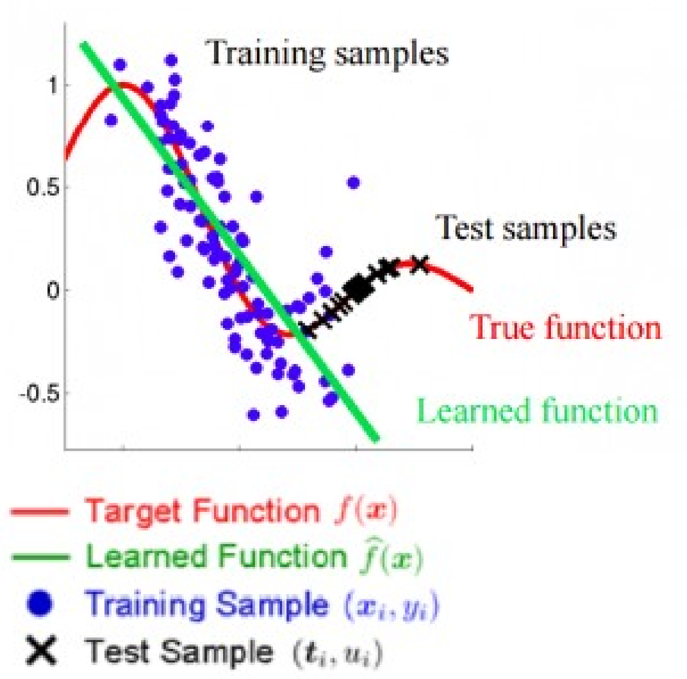
Stratified sampling aims at splitting a data set so that each split is similar with respect to something. In a classification setting, it is often chosen to ensure that the train and test sets have approximately the same percentage of samples of each target class as the complete dataset.

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=[9000,100], centers=None, cluster_std=[10.0, 2],random_state=22,n_features=2)

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.10,stratify=y,random_state=42)

print(f'Percentage of minority class in test={((y_test==1).sum()/len(y_test))*100:.3f}')
print(f'Percentage of minority class in train={((y_train==1).sum()/len(y_train))*100:.3f}')
```

```
Percentage of minority class in test=1.099
Percentage of minority class in train=1.099
```

And even if you do everything right

Dataset shift often occurs when data is collected over a long period of time. Without careful curation, your training set may no longer be predictive of your test set. This is especially common in temporal datasets.



*https://www.analyticsvidhya.com/blog/2017/07/covariate-shift-the-hidden-problem-of-real-world-data-science

Summary

Calculate all metrics on train set only

- Use these metrics to normalize test set(mean, std_dev)

Dealing with unbalanced datasets

Use stratified splits for classification

Dataset shift- it’s probably going to happen