**DS Grad Programme 6 (Machine Learning – Part Two) –   
Statistical Bias**

What do we mean by statistical bias?

* When models are incorrect, there is typically no warning. They fail “silently”.
* A way that models can fail silently is by having bias within them that produces either directly “incorrect” results, or unfair / inequitable results.
* These biases can be found and removed by using a robust methodology and by taking precautions.
* If the model will not tell us what’s wrong with it, we must look for it ourselves.

Data Leakage

What is it?

* Leakage occurs when some information (such as data) is given to a model that it should not have, or will not have, at prediction time.
* Leakage can happen in a range of steps in the model creation process. It occurs as a result of how the code is written, but is rarely explicit.
* There are broadly two different types of leakage possible in Machine Learning models:
  + Training Example Leakage (think: rows of the data set)
  + Feature Leakage (think: columns of the data set)

Training Example Leakage

* An important part of predictive modelling is being able to evaluate the model’s performance - we want our models to be accurate and to capture the patterns and processes within our data.
* To reduce bias, we use a data set that the model was not trained with to measure how accurate the model’s predictions are.
  + The performance of a model is largely checked by this held-out set of data points.
  + If information from the test data set is shared with the model then the estimate of the model performance will be biased.
* Common examples of training example leakage
  + Premature Featurisation
    - When we are exploring how to design our model we will make decisions based on the data we look at.
    - If we make decisions based off of the whole data set, rather than just the data the model will be trained on, then we are using information about the test data to create the model.
    - It is therefore crucial that we split our data into training and test sets before we explore or create features for the model.
    - Example
      * We want to create a new feature based on the “mean radius” column of our data set. We are going to standardise the data, by subtracting the mean and dividing by the standard deviation for each record.
      * To do this we need to calculate two summary statistics: the mean of the data, and the standard deviation.
      * In our processing we therefore should calculate the mean and standard deviation from the training data, then use those values on the test data. If we calculate the mean and standard deviation using both the training and test data then the samples from the test data will influence the values calculated.
    - How to avoid this problem
      * Split Early - ensure you have a hold out data set before any exploration, ideally stored separately from the training data.
      * Be careful - peer review the work and specifically looking for data leakage can help avoid it.
      * Use pipelines - modules for most modern machine learning libraries allow you to combine steps in a workflow that helps prevent accidental leakage.
    - ***Key quality questions***
      * What steps have been taken to avoid data leakage?
      * How “held out” was the test/validation set?
      * At what stage is the data split, and when are the features created?
  + Duplication
    - It is tempting to create new instances of existing data points to “re-balance” the data set, or to “augment” the examples by duplicating and then altering the duplicates some way.
      * Missing duplicates - Forgetting to check training data for duplicates
      * Over sampling - if we have lots of class A, and few instances of class B, we may be tempted to repeat the samples of class B until the data is more balanced between A and B.
      * Data Augmentation - if we have few samples of images to classify with, we may repeat images each with a rotation, skew or some other transformation to help our model learn.
    - If we create duplicates or augment our data set before we split our training and test sets, then identical or similar instances will exist in both the training and test data sets.
      * This will skew our evaluation of the model, as the model will have already seen the data during training that is subsequently used to measure its performance.
    - ***Key quality questions***
      * Has data been augmented / created and where in the process does this happen?
      * Is the model evaluated using duplicate samples?
  + Independent Identically Distributed Data
    - A key assumption we make when evaluating the performance of a model is that the data it was trained with is:
      * Independent of the other samples
      * Identically Distributed - the samples come from the same population
      * When these assumptions are not held data leakage can easily occur.
    - Time series data is particularly susceptible to this issue, as data points may depend on previous data points (eg. ice cream sales being determined by whether it was sunny yesterday or not)
      * The data is not independent of other samples
        + If we split up the data at random we are not following the assumption of independence.
      * There will be trends (e.g a decline in sales over time for example) that will not be captured, this means that each sample is not identically distributed either, the population is moving over time.
    - Instead of randomly splitting our time series data, we should train the model on past data, then test it on a held out newer data set.
    - ***Key quality questions***
      * Are the samples independent?
      * Are the samples from the same distribution?
      * Is a random split appropriate for this evaluation?

Feature Leakage

Feature leakage happens when we give a model a feature that it shouldn’t have been able to see. These are generally:

* Including the target in the data set (giving it an X which includes what we are predicting y) – easy to do!
* Including a proxy for the target in the data set
  + If we have one feature that is disproportionately doing the “heavy lifting” of the prediction it may be a proxy for the thing we are trying to predict.
  + When we discuss proxies in the context of feature leakage we are referring to data that is directly related to the outcome, such as a restructuring of the outcome.
* ***Key quality questions***
  + What are the features of the model?
  + Could any of the features be proxies for the target?
  + Do any of the features account for a substantial part of the prediction?

Sample Imbalance

Nature of the problem

* Our model can only predict based on what it has seen
  + If it has seen more of a certain type of data point, it will naturally think that that data point is more likely to occur (the mechanism of this depends on the model).
* To improve our models performance for different breakdowns we may need to make our training data less representative of the overall population, to make it better at predicting for that same population.
* The imbalance issues often arise when quantifying the target/label/value we are trying to predict, however, this can be weakly transferred to any feature in the data set.

Effects of class imbalances

* Broadly we say that as the class imbalance increases (difference in proportion) the performance of prediction for the less common class decreases.
* We therefore need to strike a balance between improving the performance of one class with the other.
* If we have imbalanced classes and are not taking them into account, we are setting up the model for failure, which will, as we saw in the last module, disproportionately affect some groups present in our data.
* ***Key quality questions***
  + How imbalanced is the phenomena being modeled?
  + How imbalanced was the training data used to create this model?

Mitigating class imbalances

* Gather More Data (preferred option)
* Re-Sampling
  + When discussing re-sampling, and any row-based manipulation of the training data we need to keep in mind how we avoid data leakage, as discussed in the section on clustering.
  + Oversampling
    - The first approach is to produce new data points from the class with the least data.
      * The most naive approach is to create duplicate data points by sampling the class with the least data until some required count is reached.
      * This can improve model performance, but it’s important to remember that this will add bias into the data set. If the class with the least data was not very representative, by oversampling we propagate that problem further.
    - Better to use a method such as SMOTE
  + Undersampling
    - This approach creates balance in the data set by randomly removing data points from the majority classes until they have the same number of data points as the smaller classes.
    - This however removes a lot of useful information from our model with the hope of re-balancing the data set.
    - This approach is most useful when we know we have a very representative sample of the majority class already, and that we do not want to bias the model by oversampling the larger class.
    - As discussed previously, we tend to want more data, rather than less. However, in comparison to oversampling this doesn’t add bias as explicitly.
    - Can also use more advanced methods such as SMOTE or Tomek
    - ***Key quality questions***
      * Could over/under-sampling reduce bias for certain classes?
      * Is there enough data to perform sample based balancing?
* Weighting
  + Not all data is created equally, therefore our model’s shouldn’t need to treat every data point equally.
  + By re-weighting the classes a model sees we can re-balance the predictions.
    - This in effect penalizes different prediction errors by different amounts.
  + Getting a prediction of a smaller class is much worse for our re-balancing than incorrectly predicting a larger class.
    - We can decide how much we want to re-weight the classes in our data set, depending on what our goals are.
  + Not all model types allow us to re-weight the classes, those that can include:
    - linear models, decision trees, support vector machines.
  + ***Key quality questions***
    - Are the models classes / outcomes appropriately weighted?
    - If there is significant class imbalance in the original data, what steps have been taken to address this?

Validation

Three Way Holdout

* If we want to evaluate a model on unseen data, we can’t have used that data at any step previously to producing the model we want to evaluate.
* Solution is to do a three way holdout
  + A training set - used to train a model
  + A validation set - used to improve the model, decide how to optimize
  + A test set - used only at the end of the design process to evaluate performance
  + Typically, the data will be split along the lines of: Training (60%), Validation (20%) and Test (20%). If we do not have enough data to have representative samples in each set then other methods may be needed.
  + ***Key quality question***
    - How much has the test data set been used to influence the model design?

Cross Validation (K-Fold Cross Validation)

* The most common type of cross validation is called K-fold cross validation, where K is an integer.
* The data set is randomly sampled into K equally size subsets.
* The model is trained K times, on K-1 combined sets, and evaluated on the other set.
* This way we get K different models, that have been evaluated on every individual point in the data set.
* Each model trained is trained on slightly different sets of data, which will cause each model to be slightly different. How different they are is our variance. As each model is evaluated on a different test set, we get a broader understanding of the performance of the model, rather than just one single evaluation.
* ***Key quality question***
  + How much does the model vary with different input data?

Stratification

* Even if we have a robust balance in the training set, when we cross validate on this set we are completely dependent on random sampling to preserve whatever balance we have constructed.
* The act of stratifying our data set ensures we maintain a balance of classes when we split the data, such as in each k-fold in cross validation.
  + Particularly important when our folds becomes small, as we are more likely to have unrepresentative samples for each fold.
    - If we have folds that are unrepresentative of our overall class balance we will have artificially high variance in our cross-validated model evaluation.
    - Any aggregate we perform on the performance of the folds will be skewed by this effect.
* ***Key quality question***
  + Are the data sets being evaluated with appropriately distributed?

Data Set Size

* The approach of data splitting and evaluation is largely depended on how much data is available.
  + When the data sets are smaller, we need to make each point count more, while still not biasing our evaluation.
  + ***Key quality question***
    - Has the data been separated for training and testing in an appropriate way?

Model Hyperparameter Optimisation

Some common hyper-parameters (with many more specific to each model):

* Regularization coefficients (how much to penalise high coefficient values)
* Solving algorithm (which approach to solving an optimisation problem)
* Solving algorithm tolerance threshold (at what point to stop solving the optimisation problem)
* Class weights (the balance of each target value)

Searching for optimal hyper-parameters is an occasional double-edged sword.

* Because of the flexibility of our algorithms we can select better hyper-parameters for the task at hand, great.
* However, this introduces the possibility that we overfit the model to the specific data set.
* So, the flexibility of the model needs to be balanced.
  + If we select a set of hyper-parameters that perform really well on our training data, they may not perform well on our test data.
  + Remember: we care about the performance on a held out set because it is a better approximation of real data the model will see in practice.
* Over-fitting happens when our model captures signal and noise, whereas we only want signal. Tips to avoiding overfitting:
  + Keep models simple
  + Evaluate robustly on held-out data
  + Search through a variety of hyper-parameters
  + Provide adequate data
  + Regularize coefficients (or early stopping/dropout for deep learning)
  + Use cross-validation methods when searching for the best hyper-parameters
* ***Key quality question***
  + What investigations into appropriate hyper-parameters have been conducted?