**KEYWORDS**

Repository/repo: ***folder*** where we are going to be storing files in GitHub

Forking: forking a repository is GitHub lingo for ***making a copy***

Feature/Attribute of a dataset: a COLUMN in a dataset

NaN: an object that comes from numpy library. Indicates missing value. NaN has datatype of “float”

Casting: changing something’s datatype –casting from an integer to a string

Series (capitalized): the technical name for a column of a dataframe

Date/time objects (for Pandas):

*Set* - a collection of unique entities, where any given entity is either contained or not contained in the set.

*Subset* - if a set AA is such that all its members are also members of another set BB then it is a subset of $B$.

*Empty set* - a set without any members. It can be defined as the set that is the subset of every set and every set (including itself) is a subset of itself. If that’s a bit mind-bending that’s OK, it’ll sink in.

*Universal set* - the set that every set is a subset of.

Permutation: sets with an order to them—can be expressed using factorials

Pieces of a survey:

* *The question* - “Which soda do you prefer, Coke or Pepsi?” Possible issues with this - by leading with Coke, we may be biasing people to select with that. A better approach is to randomize the order of the options (ballots do something similar).
* *Sampling process* - for surveys, RDD (random digit dialing) is typical. The good news is this really is random in terms of selection, but in terms of response it is decisively not random - not everyone has a landline, and not everyone answers or participates in surveys given the opportunity. The main mitigation is to also collect demographic data, and weight the responses accordingly - we won’t go into depth on this (it’s not necessary for online A/B tests), but if you get into survey design it’s an important approach.
* *Null hypothesis* - in this case, the simplest null hypothesis would be that neither Coke nor Pepsi is the favorite - that is, both are preferred by half the population.
* *Hypothesis test* - this is the approach to take the above and calculate a test statistic, which will allow us to interpret the results. One common test is [Student’s t-test](https://en.wikipedia.org/wiki/Student%27s_t-test) - it is applied in cases where the test statistic is distributed normally (a very typical situation), but the standard deviation of the population is unknown (which, for lots of social/real-world situations, is also typical). It is a relaxed version of Z-tests (which assumes more strict normality and a known standard deviation).

Aside from distribution, another important decision is selecting the type of test - *one-sample*, *two-sample*, or *paired*. One-sample tests compare results from a single sample against a known or hypothesized population. Two-sample tests compare two different samples against one another. Paired tests are a special type of two-sample tests where, to control other variables, observations are paired and the differences between those paired observations becomes the sample.

Statistical parameters: properties of the entire population

Statistics: properties of a sample of the entire population. Inferential statistics allow us to use these samples to make generalizations about the populations from which the samples were drawn

T-Test:

Null hypothesis

Alternate hypothesis

Confidence level: 95%

P-value

Conclusion: reject or fail to reject the null hypothesis. We will reject the null hypothesis whenever the p-value is less than 1 – confidence level

Algorithm: the set of instructions that we give to a computer to complete certain tasks

Data structure: the method we use to organize and store our set of instructions so that the computer can do them super fast

Computer science: the practice of optimizing the organization of our data structures so that they run quickly and efficiently

List (Python): one of the simplest ways that we can store data and maintain their order

Vectors and matrices: lists and 2D lists

Vector: a vector of dimension *n* is an ordered collection of *n* elements, which are called components. Vector notation variables are commonly written as a bold-faced lowercase letters or italicized non-bold-faced lowercase characters with an arrow above the letters.

Matrix: a matrix is a rectangular grid of numbers arranged in rows and columns. Variables that represent matrices are typically written as capital letters and sometimes boldfaced.

Scalars: numbers stored as a variable. It is a single whole or decimal number. Variable representing scalars are typically written in lower case

Dimension: the number of rows and columns that a matrix has.

* When listing the dimensions of a matrix we always list rows first and then columns.

Matrix equality: in order for two matrices to be equal the following conditions must be true:

1. They must have the same dimensions
2. Corresponding elements must be equal

Matrix multiplication: you can multiply any two matrices where the number of columns of the first matrix is equal to the number of rows of the second matrix

* Matrix multiplication is best understood in terms of the dot product: a⃗ ⋅b⃗ =(a1×b1)+(a2×b2)+…+(an×bn)
* To multiply matrices together, we will take the dot product of each row of the first matrix with each column of the second matrix. The position of the resulting entries will correspond to the row number and column number of the row and column vector that were used to find that scalar
* The number of columns in one matrix must match the number of rows in the other matrix



Transposed matrix: a transposed matrix is one whose rows are the columns of the original and whose columns are the rows of the original

* Common notation for the transpose of a matrix is to have capital T superscript or a tick mark
* A transpose of any matrix can be found easily by fixing the elements of the main diagonal and flipping the placement of all other elements across that diagonal

Square matrix: any matrix that has the same number of rows as columns

* Diagonal: values are on the main diagonal (top left to lower right), zeroes everywhere else
* Upper triangular: values on and above the main diagonal zeroes everywhere else
* Lower triangular: values on and below the main diagonal, zeroes everywhere else
* Identity matrix: a diagonal matrix with ones on the main diagonal and zeroes everywhere else (a diagonal matrix where the values are all 1)
* Symmetric: the numbers above the main diagonal are mirrored below/across the main diagonal

Determinant: a property that all square matrices possessed and denoted *det*(A) or using pipes |

* The area of the parallelogram that the vectors form
* As determinant approaches 0, the vectors become redundant



Inverse: the inverse is like the reciprocal of the matrix that was used to generate it

* The product of a matrix multiplied by its inverse is the identity of the matrix of the same dimensions as the original matrix
* Matrices that are not square are not invertible
* A matrix is invertible if and only if its determinant is non-zero because the denominator of the fraction is the determinant (*det*(A))for you

Convolving: the process of passing a filter/kernel (small matrix) over the pixels of an image, multiplying them together, and using the result to create a new matrix. The resulting matrix will be a new image that has been modified by the filter to emphasize certain qualities of an image.

Norm of a vector (magnitude or length): the norm or magnitude of a vector is nothing more than the length of the vector. Since a vector is just a line (essentially) if you treat it as the hypotenuse of a triangle you could use the Pythagorean theorem to find the equation for the norm of a vector.

* We denote the norm of a vector by wrapping it in double pipes ||v||
* The norm of a vector is the square root of the sum of the squared elements of a vector
* The norm is always positive or 0 (only if all the elements of the vector are 0)
* The equation for the norm of a vector is just a^2 + b^2 = c^2

Dot product: the dot product of two vectors is a scalar quantity that is equal to the sum of pair-wise products of the components of vectors a and b.

* a⃗ ⋅b⃗ =(a1×b1)+(a2×b2)+…+(an×bn)
* The dot product is commutative: a\*b = b\*a
* The dot product is distributive: a(b+ c) = a\*b + a\*c
* Two vectors must have the same number of components in order for the dot product to exist. If the length differs the dot product is undefined.

The first part of fixing an error is understanding it. So, let's talk about the 3 Error Types.

1. **Syntax Error** - “Syntax” refers to the structure of a program and the rules about that structure. For example, parentheses have to come in matching pairs, so (1 + 2) is legal, but 8) is a syntax error. If there is a syntax error anywhere in your program, Python displays an error message and quits, and you will not be able to run the program.
2. **Runtime Error** - An error that does not appear until after the program has started running. These errors are also called exceptions because they usually indicate that something exceptional (and bad) has happened. Both Syntax and Runtime Errors generate error messages that are helpful for understanding what went wrong and where.
3. **Semantic Error** - “Semantic” means related to meaning. If there is a semantic error in your program, it will run without generating error messages, but it will not do the right thing. It will do something else. Specifically, it will do what you told it to do. Identifying semantic errors can be tricky because it requires you to work backward by looking at the output of the program and trying to figure out what it is doing.

Predictive modeling

Heuristics: “rules of thumb” that people use to make decisions and judgments

Descriptive statistics: using data to try to do better than heuristics

Baseline:

* The score you’d get by guessing
* Fast, first models that beat guessing
* Complete, tuned “simpler” model
* Minimum performance that “matters” to go to production and benefit your employer and the people you serve
* Human-level performance

y-variable

* Dependent variable
* Response variable
* Outcome variable
* Predicted variable
* Measured variable
* Explained variable
* Label
* Target

x-variable:

* Independent variable
* Explanatory variable
* Regressor
* Covariate
* Feature

Bias-Variance Tradeoff:

* **Variance** is when a model is sensitive to noise in the training data, and generalizes poorly, causing overfitting
* **Bias** is when a model isn’t recognizing relations between features and output, causing underfitting
* Bias and variance exist in a continuum, and have a natural tradeoff—increase one and the other goes down
* **Regularization** adds bias to reduce variance. One method of regularization is Ridge Regression.

Regularization/ridge regression

* Ridge regression uses L2 regularization, which minimizes the sum of squared error *plus the sum of the squared coefficients* (times a scaling parameter, lambda)
* The result—depending on lambda, the model doesn’t want larger coefficients and gives flatter results
* As lambda approaches 0, we get to OLS—the sum of squared error (high variance)
* As lambda approaches infinite, we end up with just the mean (high bias, horizontal line)
* When **least squares** determine values for the parameters in its equation, it minimizes the sum of the squared residuals
* When **ridge regression** determine values for the parameters in its equation, it minimizes the sum of the squared residuals ***plus*** lambda times the slope(squared)
* In general, the ridge regression ***penalty*** contains all of the parameters except for the y-intercept
* While least squares needs at least as many data points as parameters, ridge regression can work with fewer data points as parameters by using **cross validation** and the **ridge regression penalty** that favors smaller parameter values
  + When sample sizes are relatively small, the ridge regression can improve predictions made from new data (i.e. reduce variance) by making the predictions less sensitive to the training data by adding the ridge regression penalty to the thing that needs to be minimized
  + The ridge regression penalty itself is lambda (determined using cross validation) times the sum of all squared parameters, except for the y-intercept
    - Lambda x slope(squared)
* We become suspicious as the ols approaches 0
* Alpha = lambda
  + The **alpha** parameter corresponds to the weight being given to the extra penalty being calculated
  + As **alpha** increases, we give more and more penalty to a steep slope
  + Ridge regression minimizes the sum of square error of the residuals **and** the squared slope of the fit model, times the alpha parameter

One hot encoding: adds a dimension for each unique value of each categorical feature, not great for features that have many unique values (“high cardinality”)

Feature engineering: the process of using domain knowledge of the data to create features that make machine learning algorithms work

* Can include making new features or representing features in new ways
* Also includes feature selection (choosing which features or include or exclude)

Well-posed problem (math) is one for which:

* A solution exists
* The solution is unique (necessary for linear algebra and system of equations)
* The solution’s behavior changes continuously with the initial condition

Ill-posed problem (math) is one for which:

* No solution exists
* There are multiple solutions
* Chaotic systems—unpredictable

Logistic Regression: limits the full output range to something that lives in the unit interval

* “Squishification” refers to using a **link function** that maps values from the real line to the unit interval
* This allows us to fit a regression that, at its heart, is linear (i.e. the underlying math/optimization), but has a non-linear boundary/restriction on its output
* From here, we can model probabilities, binary values, and even general (multinomial) classification problems
* Predicts true or false, fitting an S-shaped logistic function
* Logistic regression is able to provide probabilities and classify new samples using continuous and discrete measurements
* One big difference between linear and logistic regression is how the line is fit to the data
  + Linear regression uses least squares to find the line that minimizes the sum of the squares of these residuals
  + We also use the residuals to calculate R^2 and to compare simple models to complicated models
  + Logistic regression doesn’t have the same concept of a residual, so it can’t use least squares and it can’t calculate R^2
  + Instead, it uses something called maximum likelihood

Train/validate/test:

* The train set is for fitting the model
* The validation set is for adjusting a model’s hyperparameters
  + To estimate prediction error for model selection
* The testing data is the ultimate judge of model performance
  + For assessment of the generalization error of the final chosen model

Majority classifier: for classification tasks, a good baseline is the majority classifier, a naïve classifier that always chooses the majority class of the training dataset

Accuracy: a classification metric, it is the proportion of correct classifications