**Machine Learning Definition**

**Supervised Learning** – Uses historical and labeled data, the ML model predicts a value

* Required historical data
  + Known results and data from the past
* Labeled
  + The desired output is known
* Two main labels typed
  + Categorical Value to Predict
    - Classification Task
      * Predict an assigned category
        + Cancerous vs benign tumor
        + Fulfillment vs credit default
        + Handwriting Recognition
  + Continuous Value to Predict
    - Regression Task
      * Future Prices
      * Electricity loads
      * Test Scores

**Unsupervised Learning** – Uses unlabeled data, the ML model discovers possible patterns in the data.

* Group and interpret data without a label
  + Clustering customers into separate groups based off their behavior
* Major downside is no historical data

**Supervised Learning**

Predicting an Outcome

**Process**

**Selling a home data chart has Area Bedrooms Bathrooms and Price**

**Features-Area, Bedrooms, Bathrooms**

**Label- Price**

Data-> X: Features Y: Label-> Training Data Set (Split) 70% -> Test Data Set (Split) 30% -> Fit/Train Model -> Evaluate Performance -> Deploy Model

**Linear Regression 3.1**

Y = mx + b – slope of a line

**Ordinary Least Squares** – works by minimizing the sum of the squares of the differences between the observed dependent variable (values of the variable being observed) in the given dataset and those predicted by the linear function.

**Mean Absolute Error** – Mean (avg) of the absolute value of the errors. This may miss one or two values where there is a big difference while the majority of the values are close. Larger errors are not “punished” as much

**Mean Squared Error** – Larger errors are “punished” more than with MAE making MSE more popular. Squaring it will make the issue larger. Issue is differences are by the units being squared.

**Root Mean Square Error** – This is the most popular and will take the square root of the values after the units are resolved. This has the same units as y.

**Polynomial Regression**

* Creates the bias (value of 1.0)
* Values raised to a power of degree ( x^1 x^2 x^3)
* Interactions between all pairs of features ( x1\*x2, x1\*x3)

Ex.

* A&B: 1, A, B, A^2, AB, B^2
* X=2, y=3: 1, 2, 3, 4, 6, 9

**Non-linear Relationships** – ex is y=log(x) <- not linear.

**Sync with multiple features** – creating a new feature by multiplying two together.

**Bias-Variance Trade Off** – Underfitting or overfitting models

**Overfitting** – This model fits too much to the noise from the training data. Often results in low error on training sets but high error on test/validation sets. Harder to detect because it performs good on training data but poorly on all data.

**Underfitting** – Does not capture the underlying trend of the data and does not fit the data well enough. Low variance but high bias. Underfitting is often a result of an excessively simple model. Generalizing too much. Can lead to poor performance in both training and test data sets.

**Regularization** – Seeks to solve a few common model issues. A way to reduce overfitting and variance. All about adding parameter hyper variance. Guessing and checking the parameter.

* Minimizing model complexity
* Penalizing the loss function
* Reducing model overfitting (add more bias to reduce model variance)

**3 Types of Regularization**

* L1 Regularization – adds a penalty equal to the absolute value of the magnitude of coefficients
  + LASSO Regression
* L2 Regularization – adds penalty that equals the square of the magnitude of coefficients.
  + Ridge Regression
* Combining L1 and L2 – combines L1 and L2 in addition of an alpha parameter. A=0, absolute is cancelled out. A=1, not considering the square(L2)
  + Elastic Net

**Feature Scaling**

Benefit

* Improves the convergence of steepest descent algos which don’t possess the property of scale invariance.
* If features are on different scales certain weights may update faster than others, they play a role in the weight updates.
* Scaling the features allows us to directly compare model coefficients to each other even when they aren’t in the same measurements.
* Great increases in performance
* Needed for some models
* No “real” downside to scaling features

Caveats

* Must scale unseen data before feeding to model
* Can make interpretability of it more difficult.
  + Easier to compare coefficients but harder to relate back to original unscaled feature

**Two ways to scale data**

* Standardization or “Z-score normalization”
  + Rescaled data to have a mean(u) of 0 and standard deviation(o) of 1
  + Xchanged = X-u / o
* Normalization
  + Rescales all data values to be between 0 – 1
  + Xchanged = X -Xmin / Xmax – Xmin

**Fit() call** – calculates the necessary statistics (Xmin, Xmax, mean, standard deviation)

* Fit to the training data
* Calculating statistics from the full data leads to some info of the set leaking into the training upon the transform conversion.

**Transform() call** – actually scales data and returns the new scaled version of the data

**Feature Training Process**

* Train/test split
* Fit to training feature data
* Transform training feature data
* Transform test feature data