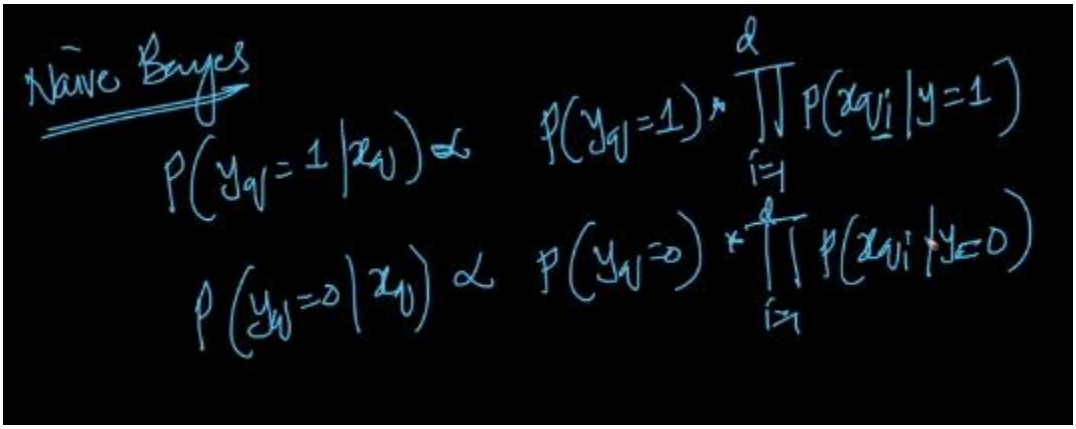
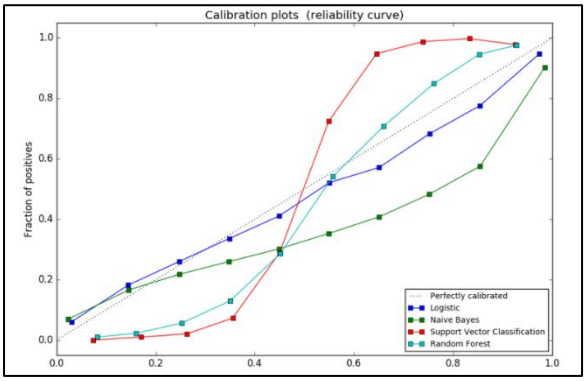
- **Calibration models**
  - Instead of predicting class values directly for a classification problem, it can be convenient to predict the probability of an observation belonging to each possible class.
  - Predicting probabilities allows some flexibility including deciding how to interpret the probabilities, presenting predictions with uncertainty, and providing more nuanced ways to evaluate the skill of the model.
  - Predicted probabilities that match the expected distribution of probabilities for each class are referred to as calibrated.
  - Calibration is a must if we want a probabilistic class-label as output i.e., $P(Y_i|X_I)$ as calibration corrects these probabilities. If we are using a log-loss as a performance metric which needs the $P(Y_i|X_I)$ values, calibration is a must.



  - This is what we have already seen in naïve bayes. And we also know that these are not actual probabilities these values are just proportional to probability.
  - For problems such as computing log loss functions which depends on actual probabilities.
  - Hence, we need to build another calibration model on top of the main model which will give us actual probability values of query point belonging to specific class.
- **Calibration Plots**
  - If model f trained on a training dataset: For each data point this model will give an output : $\hat{y}_i = f(x_i)$
  - Construct a data frame of $x_i, y_{actual}, \hat{y}_i$
  - Then we sort this data frame in increasing order of $\hat{y}_i$
  - Break table into chunks of datapoints size m i.e., for a dataset where total number of points is n after splitting the dataset each chunk will have size m
  - In each chunk compute average of y_{pred} and y_{actual} in the chunk; $\bar{y}_{pred}$ and $\bar{y}_{actual}$
  - For each of the chunk, we will get a pair of $(\bar{y}_{hat}{}^i, \bar{y}_{actual}{}^i)$
  - For building a calibration model, above data frame will be the input data and our task will be to calculate probability of that point belonging to predicted class

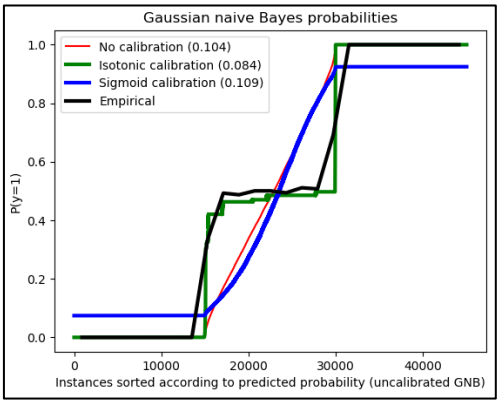


  - e.g., for m1(size = 100), 30 points belongs to class 1 & 70 belongs to class 0. The average value will be a probability of point belonging to class 1 =0.3
  - If we plot a graph with pair $(y_{hat}{}^i, y_{actual}{}^i)$ we get a calibration plot
  - Ideally this should be a line of 45°.
  - Ultimately to calibrate this model we can train these data points $(y_{hat}{}^i, y_{actual}{}^i)$ by using logistic regression

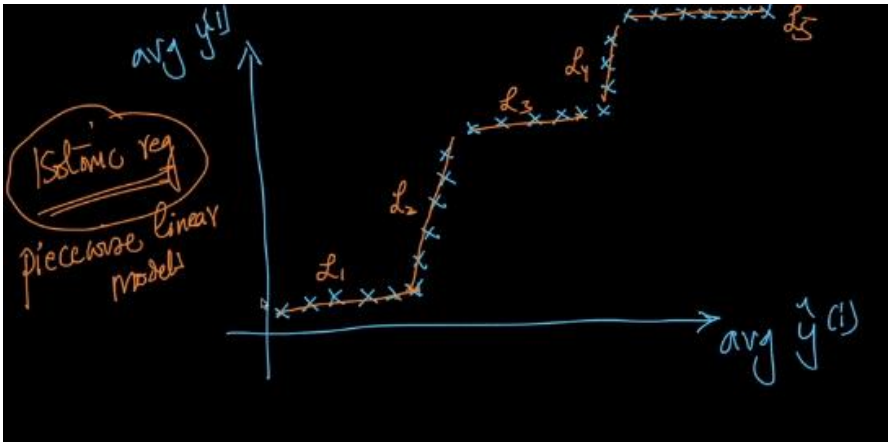
- **Platt's Calibration / Scaling / Sigmoid Calibration**
  - If we observe in above calibration graph some of them looks like sigmoid graph

$$P(y_q = 1 / x_q) = \frac{1}{(1 + \exp(A f(x_q) + B)} \mid f(x_q) = \hat{y}$$

  - Platt scaling works only if the Calibration dataset distribution is like sigmoid Function
  - For any calibration graph we need to solve above optimization problem to find out best g(X) which gives best values of A & B which will map both probabilities
  - If the calibration plot looks like following black line, it doesn't work



  - General procedure of calibration
    1. We first build a model on D_{train} and now we do cross validation on D_{cv} and get $\hat{y}_i$.
    2. Now we make a table of sorted yi_{hats} which contains actual yi's as well from the D_{cv}.
    3. We calculate the average $\hat{y}_i$ and average yi. this pair of $\hat{y}_i$ and yi is the D_{calib}.
    4. We plot the D_{calib} with $\hat{y}_i$ on x axis and yi on y axis. We want our plot to be a 45-degree line to the x axis which is the ideal situation and is generally not possible.
    5. We see that the x- axis covers the values of the $\hat{y}_i$ i.e., the predicted real value out of D_{cv} and the y axis has the actual yi's average which is nothing but the probability value. And that's what we want to predict for our test data as well.
    6. We want to give the probability value of whatever prediction we make through our model but the probability that we get directly from our base models is not the actual probability value so we make use of calibration.
    7. As using the D_{cv} we have created a model which smooths the look up table and gives a value on y axis (the probability value of yi=1) for every $\hat{y}_i$ so our calibration model knows that for which value of the predicted $\hat{y}_i$ what true value is i.e., the probability of yi
    8. so, when we give xq to the base model it predicts $\hat{y}_q$ for us and now we put this yq value in our calibration model to get the probability value of yq=1. And this is more precise and better value than the probability value that we got through our base model.
- **Isotonic Regression**
  - It works even if calibration plot does not look like sigmoid function; if we fit a sigmoid function there will be a large error
  - We break the model into multiple linear regression models which is called as piece wise linear models

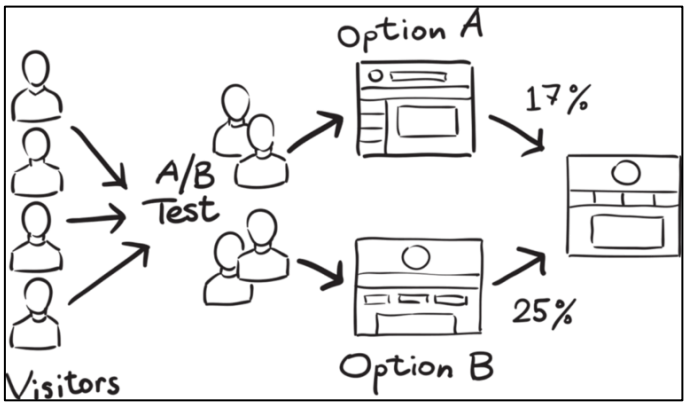


  - In order to find out these lines we need to find the threshold values $a_1, a_2, a_3, \ldots, a_n$. The connecting line between a_1, & a_2 will have a corresponding slope m_1 & intercept b_1
  - We have to Solve an optimization which finds this threshold values minimize the gap between the gaps of the model and the actual value
  - Large data → large Cross Validation dataset → large Calibration data → Isotonic Reg

- Small data → Platt Scaling

- **Code Samples**
  - Links: http://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html
  - http://scikit-learn.org/stable/auto_examples/calibration/plot_calibration.html#sphx-glr-auto-examples-calibration-plot-calibration-py
- **Random sample consensus (RANSAC)**
  - RANSAC is used to build a robust model in the presence of outliers
  - Instead of training the model on whole data we take a sample and train the dataset. Sampling will reduce the presence of outliers in the training set
  - Then we apply the model on this sample and compute loss for individual datapoints.
  - The points having high loss will be outliers
  - Determine outliers based on loss function
  - For the next iteration the dataset will be ransom sample- the outliers from previous iteration
  - Repeat the process until two consecutive models give similar results
  - Reduction of dataset at each stage with removal of outliers based on random sampling, model training and loss computation
- **Retraining models periodically**
  - For the time series datasets, if we train a model on n^{th} day it might be giving less performance metric on $(n+i)^{th}$ day
  - In that case model need to be retrained periodically to gather current trends and to get around with new companies, etc.
  - If the Dataset changes and it's dropping model performance, we must retrain model regularly
  - Such data is called as non stationary data
  - To check whether the distribution is changing or not
    - D_m = D_n (xi, yi) – (xi', yi')
      D_{n+i} (xi, yi) – (xi', yi')
    - D_m' = D_n ((xi, yi), 1)
      D_{n+i} ((xi, yi), 0)
    - Train a binary classifier on D_m'; if the accuracy is high then D_n and D_{n+i} are dis-similar; we will have small accuracy if D_n and D_{n+i} are similar; to get D_n and $D_{n+i\ i}$ follow same distribution (stable data) we will need to get small accuracy

- **A/B Testing Bucket testing or split run or controlled experiments**



  - A/B testing is a basic randomized control experiment. It is a way to compare the two versions of a variable to find out which performs better in a controlled environment.
  - In ML context, we randomly split the data into two parts A & B evenly or unevenly and then we compute some probabilistic measure on this model. E.g., confidence interval, P-Value, etc. and then we decide which model to use further
  - Just like this there is AAB model in which we fit the model twice on old model and once on a new data. The probabilistic metric additionally will tell if only the data is rearranged whether the old model is good or not.
- **Data Science Life Cycle**
  1. Understand business requirements: define the problem and the customer
  2. Data acquisition: ETL (extract transform and load), SQL [Database, Data Warehouse, Log Files, Hadoop/ Spark]
  3. Data preparation: Cleaning and pre-processing
  4. EDA: plots, visualization, hypothesis testing, data slicing
  5. Modelling, evaluation, and interpretation
  6. Communicate results: clear and simple, 1-pager and 6 pagers
  7. Deployment
  8. Real-world testing: A/B testing
  9. Customer/ business buy-in
  10. Operations: retrain models, handle failures, process
  11. Optimization: improve models, more data, more features, optimize code