I. Naive Bayes Classifier

Preparation Video:

https://www.youtube.com/watch?v=mlumJPFvooQ&list=PLZoTAELRMXVPkl7oRvzyNnyj1H
S4wt2K-&index=2&ab_channel=KrishNaik

Theoretical Understanding:

1. Tutorial 48th : https://www.youtube.com/watch?v=jS1CKhALUBQ
2. Tutorial 49th: https://www.youtube.com/watch?v=temQ8mHpe3k

1. What Are the Basic Assumptions?
Features Are Independent

2. Advantages

1. Work Very well with many number of features

   Explanation: Because of the class independence assumption, naive Bayes classifiers can quickly learn to use high dimensional features with limited training data compared to more sophisticated methods. This can be useful in situations where the dataset is small compared to the number of features, such as images or texts. Naive Bayes implicitly treats all features as being independent of one another, and therefore the sorts of curse-of-dimensionality problems which typically rear their head when dealing with high-dimensional data do not apply.

   If your data has k dimensions, then a fully general ML algorithm which attempts to learn all possible correlations between these features has to deal with 2^k possible feature interactions, and therefore needs on the order of 2^k many data points to be performant. However because Naive Bayes assumes independence between features, it only needs on the order of k many data points, exponentially fewer.

   However this comes at the cost of only being able to capture much simpler mappings between the input variables and the output class, and as such Naive Bayes could never compete with something like a large neural network trained on a large dataset when it comes to tasks like image recognition, although it might perform better on *very* small datasets.

2. Works Well with Large training Dataset
3. It converges faster when we are training the model

   Explanation: Unlike other machine learning models, naive bayes require little to no training. When trying to make a prediction that involves multiple features, we simply use the maths by making the *naive* assumption that the features are independent.

4. It also performs well with categorical features

   Explanation: For a <u>Naive Bayes</u> classifier, categorical values are the easiest to deal with. All you are really after is P(Feature | Class). This should be easy for the days of the week. Compute P(Monday | Class=Yes) and so on.

3. Disadvantages

1. Correlated features affects performance

4. Whether Feature Scaling is required?

No. *In fact, any Algorithm which is NOT distance based, is not affected by Feature Scaling.* As Naive Bayes algorithm is based on probability not on distance, so it doesn't require feature scaling.

5. Impact of Missing Values?

Naive Bayes can handle missing data. Attributes are handled separately by the algorithm at both model construction time and prediction time. As such, if a data instance has a missing value for an attribute, it can be ignored while preparing the model, and ignored when a probability is calculated for a class value tutorial : https://www.youtube.com/watch?v=EqjyLfpv5oA

6. Impact of outliers?

It is usually robust to outliers.

One potential issue with outliers is that unseen observations can lead to 0 probabilities. And we know in naive bayes we multiply probab of words lying in that particular class and results zero. For example, Bernoulli Naive Bayes applied to word features will always produce 0 probabilities when it encounters a word that wasn't seen in the training data. Outliers in this sense can be a problem.

However, all these and similar issues of Naive Bayes have well-known solutions (like Laplace smoothing, i.e. adding an artificial count for every word) and are routinely implemented. In Gaussian Naive Bayes, outliers will affect the shape of the Gaussian distribution and have the usual effects on the mean etc. So depending on your use case, it still makes sense to remove outliers.

Different Problem statement you can solve using Naive Bayes

1. Sentiment Analysis
2. Spam classification
3. twitter sentiment analysis
4. document categorization

II. **Linear Regression Algorithm**
Preparation Video:
▶ Interview Prep Day 2- Linear Regression Interview Question-The Most Importa…

Theoretical Understanding:

1. https://www.youtube.com/watch?v=1-OGRohmH2s&list=PLZoTAELRMXVPBTrWtJkn3w WQxZkmTXGwe&index=29
2. https://www.youtube.com/watch?v=5rvnlZWzox8&list=PLZoTAELRMXVPBTrWtJkn3wW QxZkmTXGwe&index=34
3. https://www.youtube.com/watch?v=NAPhUDjgG_s&list=PLZoTAELRMXVPBTrWtJkn3w WQxZkmTXGwe&index=32
4. https://www.youtube.com/watch?v=WuuyD3Yr-js&list=PLZoTAELRMXVPBTrWtJkn3wW QxZkmTXGwe&index=35
5. https://www.youtube.com/watch?v=BqzgUnrNhFM&list=PLZoTAELRMXVPBTrWtJkn3w WQxZkmTXGwe&index=33

Theoretical Concepts:

y=mx + c; m = slope, c = intercept

We need to find the best fit line. If my x is 0, then y is c i.e. Y-intercept. Within a unit change in X-axis, the change in the y-value is called slope or m. m = (y2 - y1) / (x2 - x1)

The summation of squared error should be minimised by the requisite values of m and c obtained. Here, cost function = distance b/w the best fit point and the actual point should be minimum. Thus, cost function = 1/2n $\sum$(y' - y)^2 where i ranges from 1 to m. Here, m is the number of points.

Predicted points: y' and actual points: y. But, if we just use the above equation, we might get several best fit lines, and choosing just one might be time consuming.

So, what can we do?

Example: y = x; y' = mx + c; here, we can assume the best fit line passes through origin and c = 0; thus y' = mx

Now, we can substitute x = 1, and m = 1; then y' = 1;

For x = 2, y' = 2 and y = 2 and so on. This is actually my best fit line when my slope is 1. After getting this equation, we calculate our cost function and try to reduce it.

Here, cost function, J(m) = 1/2n((1-1)^2 + (2-2)^2 + (3-3)^2) = 0

With respect to every m value, we can plot our cost function. For m = 1, J(m) = 0.

For m = 0.5, x = 1; y = 0.5,

For x= 2, y = 1,

For x = 3, y = 1.5

Then our cost function, J(m) = 1/2n((1-0.5)^2 + (2-1)^2 + (3-1.5)^2) = 0.58. Thus, a curvature of J(m) versus m can be plotted and we can see the gradient descent. How do we arrive at the global minimum?

Based on some m value, we get some initial J(m) value. In order to move downwards, we should use the convergence theorem.

Convergence theorem: $m = m - \alpha \partial J/\partial m$ where α is the learning rate

If the slope ($\partial J/\partial m$) is negative, the curve points downwards. When we get a negative slope, then m = m + (+ive smaller value). This step would be very very small and it would move slowly towards global minimum. If we take a larger alpha, then the jumps might be bigger and it might not converge after several iterations and keep oscillating.

At the global minima, the slope will be zero. This would be the slope of the best fit line. Until then, we would keep following the convergence theorem.

If I have multiple independent features, then each of the features will try to reach a global minimum.

For multiple linear regression, y = β_0 + $\beta_1 x_1$ + $\beta_2 x_2$ + $\beta_3 x_3$ + $\beta_4 x_4$ + $\beta_5 x_5$; here, β_0 is the y-intercept; β_1, β_2 and β_3 are changes in y with unit changes in x_1 , x_2 and x_3 respectively.

Multicollinearity: In regression, "multicollinearity" refers to predictors that are correlated with other predictors. Multicollinearity occurs when your model includes multiple factors that are correlated not

just to your response variable, but also to each other. In other words, it results when you have factors that are a bit redundant. To handle such a multicollinearity situation, one solution is to remove highly correlated feature (check for P value in model summary and remove the one for which it is higher).

R Square and Adjusted R Square:

$R^2 = 1 - SSres/SStot$

Here, $SSres$ is the sum of squares of residuals or errors = $\sum(y' - y)^2$

$SStot$ is the sum of average totals = $\sum(y' - y_{mean})^2$

The closer the value of R^2 to 1, the better the model is.

It will be less than zero only when the best fit line is worse than the average value.

As we go on adding new independent features, our R^2 value usually increases. This is because the model then tries to apply some coefficient value such that our $SSres$ value decreases. Then, resultantly, our R^2 value increases. We should also note that this value will never decrease when we keep on adding independent features to it. But, there might be a scenario that the independent feature being added might not be correlated to the target output. Even though R^2 value might show an increase in such a case, but it is basically not penalising the newly added features which do not have any correlation with the target. For this reason, we use adjusted R square.

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

Where
R^2 Sample R-Squared
N Total Sample Size
p Number of independent variable

From the above formula, we can note that as the value of p increases when independent features which are not correlated to the target variable are added, N - p - 1 value decreases, leading to an overall decrease in the Adjusted R square value as a higher term is subtracted from 1 to achieve it. In case, the added features have correlation with the target variable, then R^2 value would be higher such that the rest of the multiplication factor will become overwhelmed and would not make much of a difference to the overall R^2 value.

Differences between R^2 value and Adjusted R^2 value:

Every time an independent variable is added to a model, R^2 value increases, even if the independent variable is insignificant. It never declines, whereas adjusted R^2 value increases only when the independent variable is significant and affects the dependent variable.

Adjusted R^2 value is always less than or equal to R^2 value.

Interview Question on Multicollinearity

1. https://www.youtube.com/watch?v=tcaruVHXZwE

Removing the highly correlated feature would work when we have quite fewer numbers of features and a smaller dataset. We also lose certain information when we remove some data. In case, we have a large dataset, we would go for Ridge or Lasso correlation.

1. What Are the Basic Assumptions?(favourite)

There are four assumptions associated with a linear regression model:

1. *Linearity*: The relationship between X and the mean of Y is linear.
2. *Homoscedasticity*: The variance of residual is the same for any value of X.
3. *Independence*: Observations are independent of each other.
4. *Normality*: For any fixed value of X, Y is normally distributed.

2. Advantages

1. Linear regression performs exceptionally well for linearly separable data
2. Easy to implement and train the model
3. It can handle overfitting using dimensionality reduction techniques and cross validation and regularization

3. Disadvantages

1. Sometimes Lot of Feature Engineering Is required
2. If the independent features are correlated it may affect performance
3. It is often quite prone to noise and overfitting

4. Whether Feature Scaling is required?

Yes (Whenever we talk about gradient descent, we need to do feature scaling because that will help us to reach the optimum solution / global minima very quickly.)

5. Impact of Missing Values?

It is sensitive to missing values (feature engineering is used to handle it)

6. Impact of outliers?

Linear regression needs the relationship between the independent and dependent variables to be linear. It is also important to check for outliers since linear regression is sensitive to outlier effects. Just to reduce the mean squared error (MSE), the line is changing in the below figure with the appearance of an outlier. To handle it, ridge and lasso regression are used.
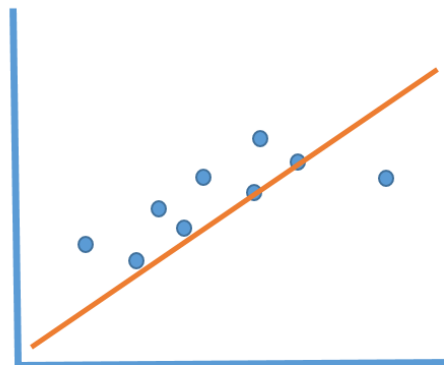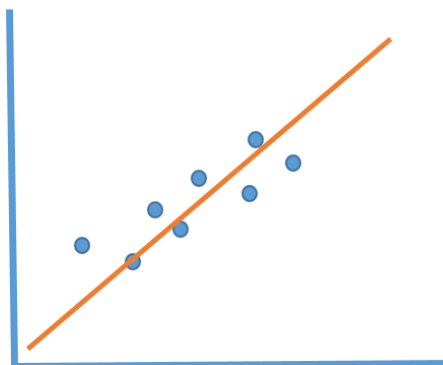
Types of Problems it can solve(Supervised)

1. Regression

Overfitting And Underfitting

We will use polynomial linear regression to explain bias and variance tradeoff as well as overfitting and underfitting. Please note that if the degree of the polynomial is one, then the curve is a straight line. The sum of mean square error (cost function) is higher in that scenario when compared to a polynomial curve with degree greater than one. When error is very high for a training dataset, then the particular scenario is called *underfitting*. Suppose, we use a quite higher order polynomial such that almost all the training data points are fitted quite well by the model, this scenario is called *overfitting*. This is due to the fact that the accuracy would go down for the test data even though it is very high for the training data.

Our main aim is to achieve an optimum solution such that accuracy is high for both training and test data. That model will give us low bias and low variance.

In an underfitting scenario, we basically have high bias (error of the training data) and high variance (generalizability - error on the test data). For overfitting scenario, we have low bias but high variance.

Different Problem statement you can solve using Linear Regression

1. Advance House Price Prediction
2. Flight Price Prediction

Practical Implementation

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

III. **Support vector Machines (SVM)**

Preparation Video:
https://www.youtube.com/watch?v=eYRBWM9Mvuc&list=PLZoTAELRMXVM0zN0cg
JrfT6TK2ypCpQdY&index=8&ab_channel=KrishNaik

Theoretical Concepts:

The main aim of SVM is to create a hyperplane and two separating margin lines such that there are two more hyperplanes in parallel to the particular hyperplane and these hyperplanes pass through the nearest neighbouring point in the two clusters.

This whole distance between the r=two parallel hyperplanes - d+ive and d-ive is called margin. We aim to get a generalised model so that it is easily applicable on unseen points. This margin and hyperplane helps in such generalisation.

Now, we can create multiple hyperplanes separating the two clusters apart from this one hyperplane. Then, why choose this hyperplane. Our main aim is to select the hyperplane which gives us the maximum marginal distance. This linear hyperplane is for linearly separable points.

What if the points are non-linearly separable?
A linearly separable hyperplane cannot be made for such a dataset. For tackling this we use kernels which convert 2 -dimension to a higher dimension. Then, between the higher dimensional dataset, we can construct linearly separable hyperplanes

What is a support vector?
The nearest positive point and nearest negative point that passes through the parallel marginal hyperplanes are called support vectors. It may have multiple support vectors.

SVM Maths Intuition:

Let us consider a simple example of logistic regression where we have two points (4,4) and (-4,0) in a 2-D plane. We wish to separate them using a linear hyperplane. Now, suppose the slope of such a linear plane is -1. The equation of this line is given as
$w'x + b = 0 \, where \, ' \, denotes \, the \, transpose.$
We also know this equation: y = mx + c; m = -1 (assumption made).
c = b = 0 as the line has been assumed to be passing through origin.Thus, $y = w'x$
$y = [-1\,0]'[-4\,0] = 4 \, (this \, value \, is \, always \, going \, to \, be \, positive)$

Anytime, we calculate y for points below the linear separator, y is going to be positive. And, when we calculate it for points above the linear separator, y is going to be negative. Now, we can take this value as + 1 and -1 for two clusters respectively.

In SVM, this hyperplane equation can also be given similar to logistic regression:
$w'x + b = 0.$
I am going to find the nearest points for the hyperplane in the two clusters. Suppose, it's at a distance -1 and + 1 respectively as assumed above.
So, the equations can be $w'x + b = -1 \, and \, w'x + b = 1$ respectively.

Now, here b will not be zero as the hyperplane is not passing through zero. Now, we basically wish to compute the distance between two points x1 and x2 lying on these two parallel hyperplanes. We can write the equations as follows:
$w'x1 + b = -1 \, and \, w'x2 + b = 1.$ So, we can subtract the two equations:
$w'(x2 - x1) = 2.$
To remove w' , we are going to divide by ||w|| on both sides. So, 2/||w|| would be our optimization function and we need to maximise this.
We need to update (w,b) to maximise the optimization function such that y = 1 when $w'x + b$ >= 1 and y = -1 when $w'x + b$ <= -1.

We can also write it as: y*$(w'x + b)$ >= 1

If it is not greater than equal to 1, then it means that it is a misclassification.

One point to note is that in a real world scenario, we will not have such separable data points. There would be a lot of overlap.

We have to change (w*, b*) such that $\min(\|w\|/2) + c\sum\zeta_i$ where i ranges from 1 to n. 'c' represents how many errors the model can consider; ζ is the value of the error. This value 'c' is called regularisation and is obtained by hyperparameter tuning. This particular SVM is called linear SVM and the margin is a **hard margin**.

(w*, b*) = $\min(\|w\|/2) + c\sum\zeta_i$ where i ranges from 1 to n

SVM Kernels:

1. Polynomial Kernels
2. RBF Kernels
3. Sigmoid Kernels

In case of soft margin, there would be certain consideration for error. Now, non-linearly separable data can be handled using the SVM kernel which will convert a lower dimensional non-linearly separable dataset into a linearly separable higher dimensional dataset. This transformation is done by SVM Kernel from lower dimension to higher dimension. It does it using some mathematical formulas.

Let us consider one-dimensional points for example. ——------*****-----****------******-----
To classify these points, in order to use SVM, we first need to draw a hyperplane but they are 1-dimensional. So, we would use a kernel to make this dataset 2-dimensional. Let us suppose, we are using a Polynomial kernel ($y = f(x) = x^2$). So, these points get projected as a parabola. Then, we can draw a hyperplane to separate these points.

Polynomial Kernel: Let us consider we have elliptical data points which are non-linearly separable in x1 and x2 plane. We will use a polynomial kernel to make the data points linearly separable in a higher dimension.

The formula for such a polynomial kernel would be: $f(x1, x2) = (x1'.x2 + 1)^d$ where d is the higher order dimension.

The unique elements in x1 and x2 multiplication would be $x1^2$ and $x2^2$ whereas the repetitive parts would be x1x2. So, we see that our 2-dimensional points are becoming higher-dimensional - x1, x2, $x1^2$, $x2^2$, x1x2 (axes: x1, x2 and x1x2) such that we can obtain linearly separable data points and hyperplane.

Theoretical Understanding:

1. https://www.youtube.com/watch?v=H9yACitf-KM
2. https://www.youtube.com/watch?v=Js3GLb1xPhc

1. What Are the Basic Assumptions?
There are no such assumptions

2. Advantages

1. SVM is more effective in high dimensional spaces. : the reason that SVMs work well with high-dimensional data is that **they are automatically regularized**, and regularization is a way to prevent overfitting with high-dimensional data.
2. SVM is relatively memory efficient. (It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.)
3. SVM's are very good when we have no idea on the data.
4. Works well with even unstructured and semi structured data like text, Images and trees.
5. The kernel trick is the real strength of SVM. With an appropriate kernel function, we can solve any complex problem.
6. SVM models have generalisation in practice, the risk of overfitting is less in SVM.

3. Disadvantages

1. More Training Time is required for larger dataset
2. It is difficult to choose a good kernel function
   https://www.youtube.com/watch?v=mTyT-oHoivA
3. The SVM hyperparameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualise their impact

4. Whether Feature Scaling is required?
Yes - Feature scaling is crucial for some machine learning algorithms, which **consider distances between observations because the distance between two observations differs for non-scaled and scaled cases**. As we've already stated, the decision boundary maximises the distance to the nearest data points from different classes.

5. Impact of Missing Values?
Although SVMs are an attractive option when constructing a classifier, SVMs do not easily accommodate missing covariate information. Similar to other prediction and classification methods, in-attention to missing data when constructing an SVM can impact the accuracy and utility of the resulting classifier.

6. Impact of outliers?
It is usually sensitive to outliers
https://arxiv.org/abs/1409.0934#:~:text=Despite%20its%20popularity%2C%20SVM%20has,causes%20the%20sensitivity%20to%20outliers.

The penalty on misclassification is defined by a convex loss called the hinge loss, and the unboundedness of the convex loss causes sensitivity to outliers.

Types of Problems it can solve(Supervised)

1. Classification
2. Regression

Overfitting And Underfitting
In SVM, to avoid overfitting, we choose a Soft Margin, instead of a Hard one i.e. we let some data points enter our margin intentionally (but we still penalise it) so that our classifier don't overfit on our training sample

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Different Problem statement you can solve using SVM

1. We can use SVM with every ANN use cases
2. Intrusion Detection
3. Handwriting Recognition

Practical Implementation

1. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

Performance Metrics

Classification

1. Confusion Matrix
2. Precision,Recall, F1 score

Regression

1. R2,Adjusted R2
2. MSE,RMSE,MAE

Which all algorithms are impacted by an imbalance dataset?

Machine Learning Algorithms such as Logistic Regression, KNN,SVM (basically which includes Gradient Descent and Euclidean Distance Computation) are affected by Imbalanced Datasets.

IV. Natural Language Processing(NLP)

Machine Learning Libraries: SpaCy, NLTK
Deep learning Libraries: PyTorch, Keras, TensorFlow
BERT, Transformers: Hugging Face Libraries