1. What is Machine Learning?

ML is the art of granting machines the ability to think. It is a field of computer science that uses statistical techniques to give machines the ability to learn without being explicitly programmed. Machine Learning deals with building algorithms that can receive input data, perform statistical analysis to predict output, and update the output as newer data become available.

2. What are some use cases of Machine Learning from our daily uses?

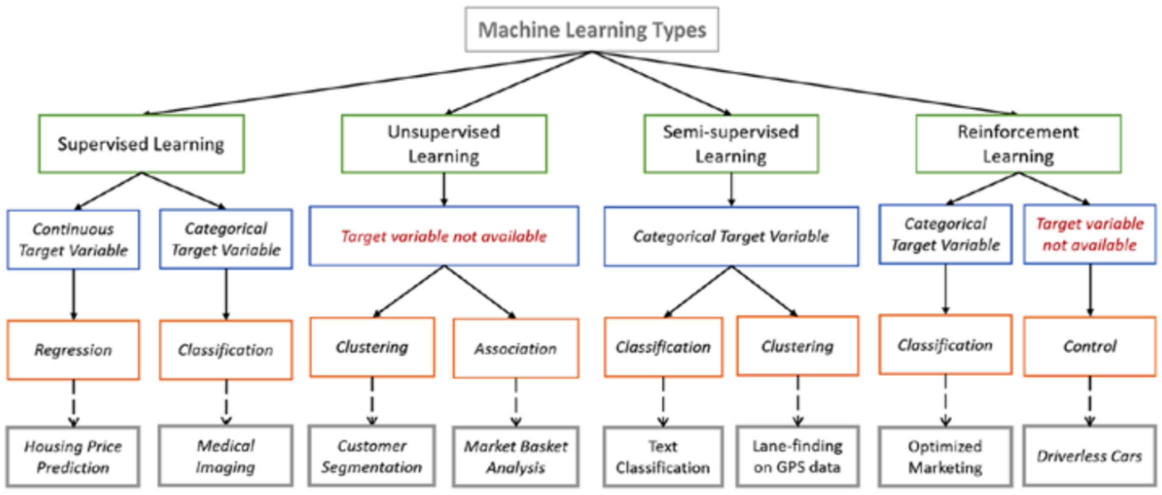
1. YouTube/Netflix/Amazon Prime: Recommends us videos or movies or series based on our interest.
2. Gmail: Classifies spam emails on your behalf
3. Banks: Detect Fraudulent/Anomalous behaviour and holds the transaction. It also helps in Fraud Detection.
4. Voice Assistance: Alexa/Siri use Machine Learning to response to the questions asked.

There are Machine Learning use cases in almost every industry today.

3. What are different kinds of Machine Learning? What does that signify?

The three kinds of learning in Machine Learning are:

1. Supervised Learning
2. Unsupervised Learning
3. Semi-supervised Learning
4. Reinforcement Learning



4. What are different steps in Machine Learning?

We perform below steps in Machine Learning:

1. Collect data
2. Filter data
3. Analyse data
4. Train algorithms
5. Test algorithms
6. Use algorithms for future predictions

5. What is Bias and Variance? How can we have optimum of both?

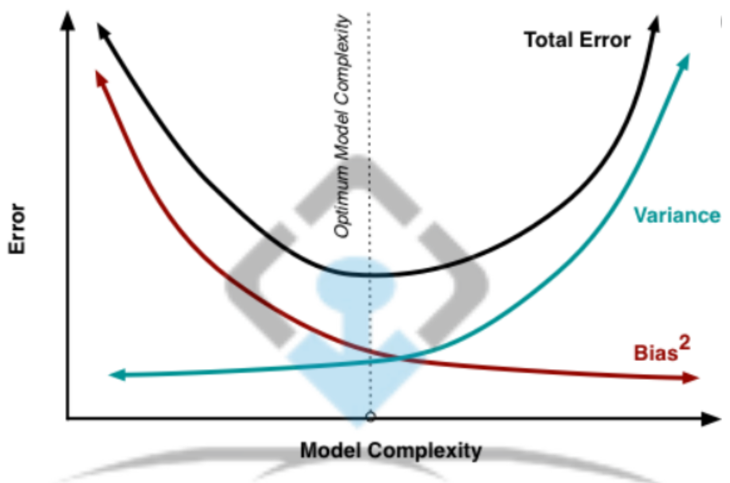
Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.

In supervised learning, underfitting happens when a model unable to capture the underlying pattern of the data. These models usually have high bias and low variance. It happens when we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data. Also, these kinds of models are very simple to capture the complex patterns in data like Linear and logistic regression.

In supervised learning, overfitting happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over noisy dataset. These models have low bias and high variance. These models are very complex like Decision trees which are prone to overfitting.

Bias Variance Trade-off:



For any supervised algorithm, having a high bias error usually means it has low variance error and vice versa. To be more specific, parametric or linear ML algorithms often have a high bias but low variance. On the other hand, non-parametric or non-linear algorithms have vice versa.

The goal of any ML model is to obtain a low variance and a low bias state, which is often a task due to the parametrization of machine learning algorithms. Common ways to achieve optimum Bias and Variance are:

a. By minimizing total error
b. Using Bagging and resampling techniques
c. Adjusting minor values in Algorithms

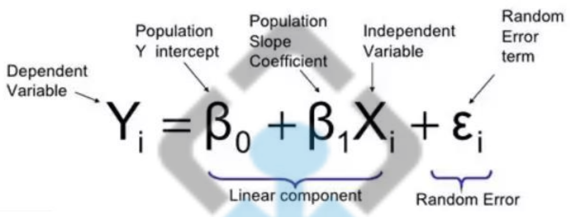
……………………………………

6. What is Linear Regression?

Linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables or independent variables.

The case of one explanatory variable is called simple linear regression, for more than one explanatory variable, the process is called multiple linear regression.

A linear regression line has an equation of the form



7. What are the assumptions of Linear Regression?

Short Trick: Assumptions can be abbreviated as LINE in order to remember.

L : Linearity (Relationship between x and y is linear)
I : Independence (Observations are independent of each other)
N : Normality (for any fixed value of x, y is normally distributd)
E : Equal Variance (homoscedasticity)

Long Answer:
There are three main assumptions in a linear regression model:

1. The assumption about the **relationship** between the dependent and independent variable: there should be a linear relationship between the dependent and independent variables. It is known as the 'linearity assumption'.

2. Assumptions about the **residuals**:

   - Normality assumption: It is assumed that the error terms, $\varepsilon^{(i)}$, are normally distributed.
     (Explanation: If the residuals are not normally distributed, their randomness is lost, which implies that the model is not able to explain the relation in the data.)
   - Zero mean assumption: It is assumed that the residuals have a mean value of zero.

   (Explanation: Zero conditional means is there which says that there are both negative and positive errorsthath cancel out on an average. This helps us to estimate the dependent variable precisely.)

   - Constant variance assumption: It is assumed that the residual terms have the same (but unknown) variance, σ^2 This assumption is also known as the assumption of homogeneity or **homoscedasticity**.
   - Independent error assumption: It is assumed that the residual terms are independent of each other, i.e. their pair-wise covariance is zero.

   (Explanation: because the observations with larger errors will have more pull or influence on the fitted model.)

3. Assumptions about the **estimators**:
   - The independent variables are measured without error.
   - The independent variables are linearly independent of each other, i.e. there is no multicollinearity in the data.

   (Explanation: If the independent variables are not linearly independent of each other, the uniqueness of the least squares solution (or normal equation solution) is lost.)

8. What is Regularization? Explain different types of Regularizations.

Regularization is a technique that is used to solve the overfitting problem of machine learning models.

The types of Regularization are as follows:

- The L1 regularization (also called Lasso)
- The L2 regularization (also called Ridge)
- The L1/L2 regularization (also called Elastic net)

L1 Regularization: L1 Regularization or Lasso Regularization adds a penalty to the error function. The penalty is the sum of the **absolute** values of weights.

Where λ is the tuning parameter called the regularization parameter which decides how much we want to penalize the model. The term for L1 regularization is highlighted below wa ith red box.

L1 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} |w_i|}$$

L2 Regularization: L2 Regularization or Ridge Regularization also adds a penalty to the error function. But the penalty here is the sum of the **squared** values of weights.

Where λ is the tuning parameter called the regularization parameter which decides how much we want to penalize the model. The term for L2 regularization is highlighted below with read box.

L2 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} w_i^2}$$

Elastic-net Regularization: Elastic-net is a mix of **both L1 and L2 regularizations**. A penalty is applied to the sum of the absolute values and the sum of the squared values:

$$\frac{\sum_{i=1}^{n}(y_i - x_i^J\hat{\beta})^2}{2n} + \lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{m}\hat{\beta}_j^2 + \alpha\sum_{j=1}^{m}|\hat{\beta}_j|\right)$$

Lambda is a shared penalization parameter while alpha sets the ratio between L1 and L2 regularization in the Elastic Net Regularization. Hence, we expect a hybrid behavior between L1 and L2 regularization.

9. How to choose the value of the regularisation parameter (λ)?

Selecting the regularisation parameter is a tricky business. If the value of λ is too high, it will lead to extremely small values of the regression coefficient β, which will lead to the model underfitting (high bias – low variance).

On the other hand, if the value of λ is 0 (very small), the model will tend to overfit the training data (low bias – high variance).

There is no proper way to select the value of λ. What you can do is have a sub-sample of data and run the algorithm multiple times on different sets. Here, the person has to decide how much variance can be tolerated. Once the user is satisfied with the variance, that value of λ can be chosen for the full dataset. One thing to be noted is that the value of λ selected here was optimal for that subset, not for the entire training data.

10. Explain gradient descent?

Definition: Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost).

(Note: Cost function is the average of the loss functions for all the training examples.)

When it is used: Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm.
Details: The goal of any Machine Learning Model is minimize the cost function. To get the minima of the cost function we use Gradient Descent Algorithm. Without going too much mathematical, let’s see the steps involved in Gradient Descent.

Step 1: Initialize the coefficient of the function. in the initial value should either be 0 a or very small value.

coefficient = 0.0

Step 2: For the coefficient, we have chosen in step 1, we will calculate the cost by putting it in function.

cost = f(coefficient)

Step 3: Find the derivative(delta) of the cost or the derivative of the function

delta = derivative(cost)

Step 4: Now that we know from the derivative which direction is downhill, we can now update the coefficient values. A Learning Rate (alpha) must be specified that controls how much the coefficients can change on each update.

coefficient = coefficient – (alpha * delta)

This process is repeated until the cost of the coefficients (cost) is 0.0 or close enough to zero to be good enough, or the number of epochs is reached.

11. What is Learning Rate? How to choose the value of the parameter learning rate (α)?

Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient. The lower the value, the slower we travel along the downward slope.

Selecting the value of learning rate is a tricky business. If the value is too small, the gradient descent algorithm takes ages to converge to the optimal solution. On the other hand, if the value of the learning rate is high, the gradient descent will overshoot the optimal solution and most likely never converge to the optimal solution.

To overcome this problem, you can try different values of alpha over a range of values and plot the cost vs the number of iterations. Then, based on the graphs, the value corresponding to the graph showing the rapid decrease can be chosen.

If you see that the cost is increasing with the number of iterations, your learning rate parameter is high and it needs to be decreased.

12. How to carry out hypothesis testing in linear regression?

Hypothesis testing can be carried out in linear regression for the following purposes:

1. To check whether a predictor is significant for the prediction of the target variable. Two common methods for this are —
   - **Using p-values**:
     If the p-value of a variable is greater than a certain limit (usually 0.05), the variable is insignificant in the prediction of the target variable.
   - **By checking the values of the regression coefficient**:
     If the value of regression coefficient corresponding to a predictor is zero, that variable is insignificant in the prediction of the target variable and has no linear relationship with it.
2. To check whether the calculated regression coefficients are good estimators of the actual coefficients.

13. What is Variance Inflation Factor (VIF)? What is its significance?

Variance Inflation Factor (VIF) is used to *detect the presence of multicollinearity*. Variance inflation factors (VIF) measure how much the variance of the estimated regression coefficients is inflated as compared to when the predictor variables are not linearly related.

It is obtained by regressing each independent variable, say X on the remaining independent variables (say Y and Z) and checking how much of it (of X) is explained by these variables.

higher the VIF, higher the R-Squared which means the variable X is collinear with Y and Z variables. If all the variables are completely orthogonal, R-Square will be 0 resulting in VIF of 1.

Rule of thumb is to avoid any variable with VIF > 5.

14. How does Residual Plot and Q-Q plot help in linear regression model?

Residual plots and Q-Q plots are used to visually check that your data meets the homoscedasticity and normality assumptions of linear regression.

A residual plot lets you see if your data appears homoscedastic. Homoscedasticity means that the residuals, the difference between the observed value and the predicted value, are equal across all values of your predictor variable. If your data are homoscedastic then you will see the points randomly scattered around the x axis. If they are not (e.g. if they form a curve, bowtie, fan etc.) then it suggests that your data doesn't meet the assumption.

Q-Q plots let you check that the data meet the assumption of normality. They compare the distribution of your data to a normal distribution by plotting the quartiles of your data against the quartiles of a normal distribution. If your data are normally distributed, then they should form an approximately straight line.

15. What is difference between R-Squared and Adjusted R Squared?

Short Answer: In case of adjusted R2 we include only the important and useful variables, whereas in R2 all the variables are included.

Long Answer: R-squared or R2 explains the degree to which your input variables explain the variation of your output / predicted variable. So, if R-square is 0.8, it means 80% of the variation in the output variable is explained by the input variables. So, in simple terms, higher the R squared, the more variation is explained by your input variables and hence better is your model.

However, the problem with R-squared is that it will either stay the same or increase with addition of more variables, even if they do not have any relationship with the output variables. This is where "Adjusted R square" comes to help. Adjusted R-square penalizes you for adding variables which do not improve your existing model. Adjusted R2 also explains the degree to which your input variables explain the variation of your output / predicted variable but adjusts for the number of terms in a model. If you add more and more **useless** variables to a model, adjusted r-squared will decrease. If you add more **useful** variables, adjusted r-squared will increase.
Adjusted R2 will always be less than or equal to R2. So, basically in adjusted R2 we include only the important and useful variables, whereas in R2 all the variables are included.

Hence, if you are building Linear regression on multiple variable, it is always suggested that you use Adjusted R-squared to judge goodness of model. In case you only have one input variable, R-square and Adjusted R squared would be exactly same.

Typically, the more non-significant variables you add into the model, the gap in R-squared and Adjusted R-squared increases.

16. What are forecast KPI's or metrics? Explain Bias RMSE, MAE, MAPE.

Forecast KPI's are used to evaluate forecast accuracy. The several KPI's for it is as follows:

Bias: **Bias** represents the historical average error. Basically, will your forecasts be on average too high (i.e. you *overshot* the demand) or too low (i.e. you *undershot* the demand)? This will give you the overall direction of the error.

$$bias = \frac{1}{n}\sum_{n} e_t$$

Mean Absolute Percentage Error MAPE is the sum of the individual absolute errors divided by the demand (each period separately). Actually, it is the average of the percentage errors.

$$MAPE = \frac{1}{n}\sum \frac{|e_t|}{d_t}$$

Issue with MAPE: MAPE divides each error individually by the demand, so it is skewed: high errors during low-demand periods will have a major impact on MAPE. Due to this, optimizing MAPE will result in a strange forecast that will most likely undershoot the demand. Just avoid it.

The **Mean Absolute Error (MAE)** is a very good KPI to measure forecast accuracy. As the name implies, it is the mean of the absolute error.

$$MAE = \frac{1}{n}\sum |e_t|$$

Issue with MAE: One of the first issues of this KPI is that it is not scaled to the average demand. If one tells you that MAE is 10 for a particular item, you cannot know if this is good or bad. If your average demand is 1000, it is of course astonishing, but if the average demand is 1, this is a very poor accuracy. To solve this, it is common to divide MAE by the average demand to get a %

$$MAE\% = \frac{\frac{1}{n}\sum |e_t|}{\frac{1}{n}\sum d_t} = \frac{\sum |e_t|}{\sum d_t}$$

RMSE: It is defined as the square root of the average squared error.

$$RMSE = \sqrt{\frac{1}{n}\sum e_t^2}$$

Issue with RMSE: Just as for MAE, RMSE is not scaled to the demand. We can then define RMSE% as such,

$$RMSE\% = \frac{\sqrt{\frac{1}{n}\sum e_t^2}}{\frac{\sum d}{n}}$$

(In all above equations, e_t is an error term)

17. What is Multicollinearity and what to do with it? How to remove multicollinearity?

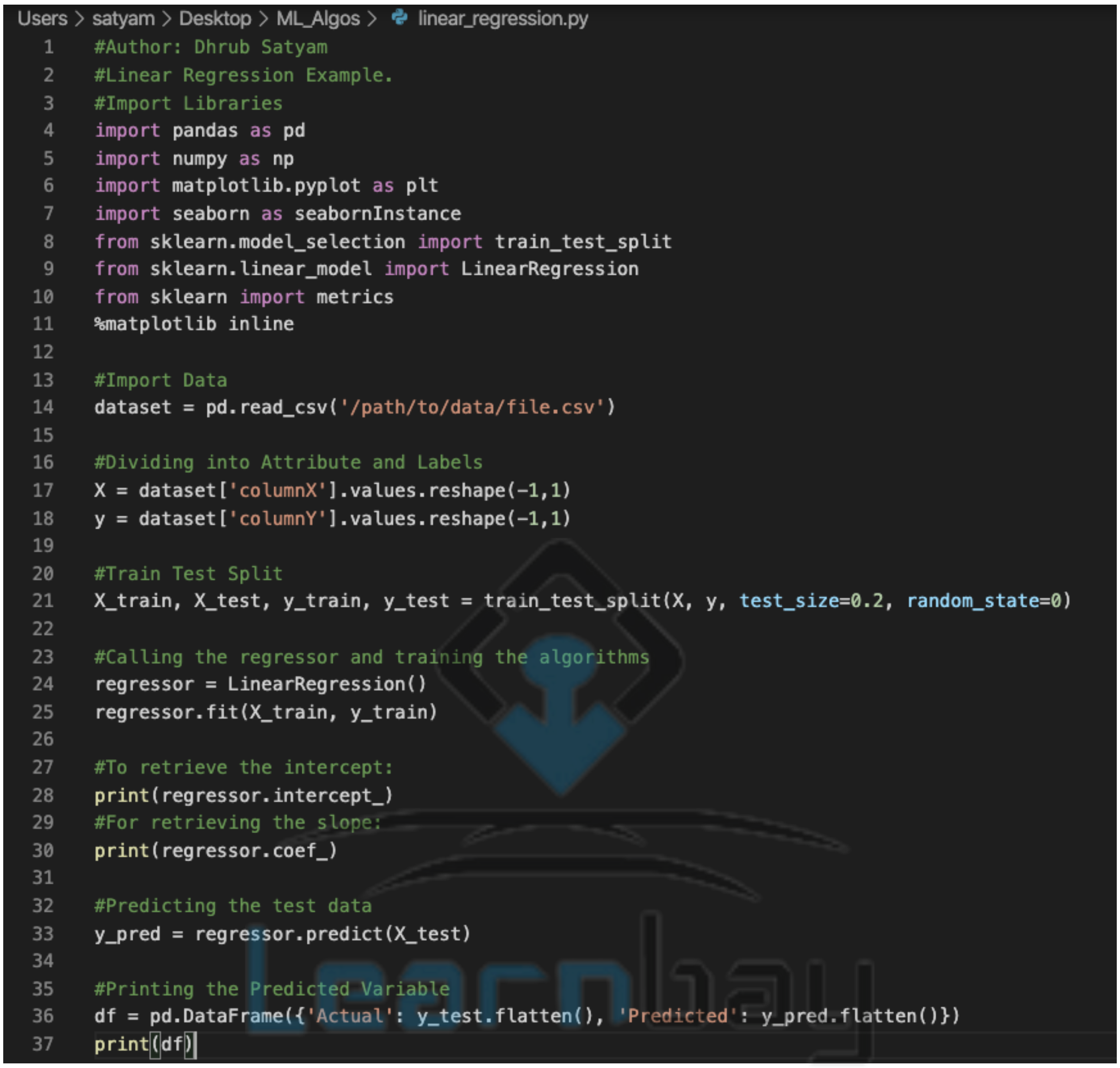
In regression, "multicollinearity" refers to predictors that are correlated with other predictors. Multicollinearity occurs when your model includes multiple factors that are correlated not just to your response variable, but also to each other. In other words, it results when you have factors that are a bit redundant.

If multicollinearity is a problem in your model -- if the VIF for a factor is near or above 5 -- the solution may be relatively simple. Try one of these:

- **Remove highly correlated predictors from the model.** If you have two or more factors with a high VIF, remove one from the model. Because they supply redundant information, removing one of the correlated factors usually doesn't drastically reduce the R-squared. Consider using stepwise regression, best subsets regression, or specialized knowledge of the data set to remove these variables. Select the model that has the highest R-squared value.

- **Use** Principal Components Analysis, regression methods that cut the number of predictors to a smaller set of uncorrelated components.

Linear Regression Code Snippet.



Users > satyam > Desktop > ML_Algos > linear_regression.py

```python
#Author: Dhrub Satyam
#Linear Regression Example.
#Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as seabornInstance
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
%matplotlib inline

#Import Data
dataset = pd.read_csv('/path/to/data/file.csv')

#Dividing into Attribute and Labels
X = dataset['columnX'].values.reshape(-1,1)
y = dataset['columnY'].values.reshape(-1,1)

#Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

#Calling the regressor and training the algorithms
regressor = LinearRegression()
regressor.fit(X_train, y_train)

#To retrieve the intercept:
print(regressor.intercept_)
#For retrieving the slope:
print(regressor.coef_)

#Predicting the test data
y_pred = regressor.predict(X_test)

#Printing the Predicted Variable
df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
print(df)
```

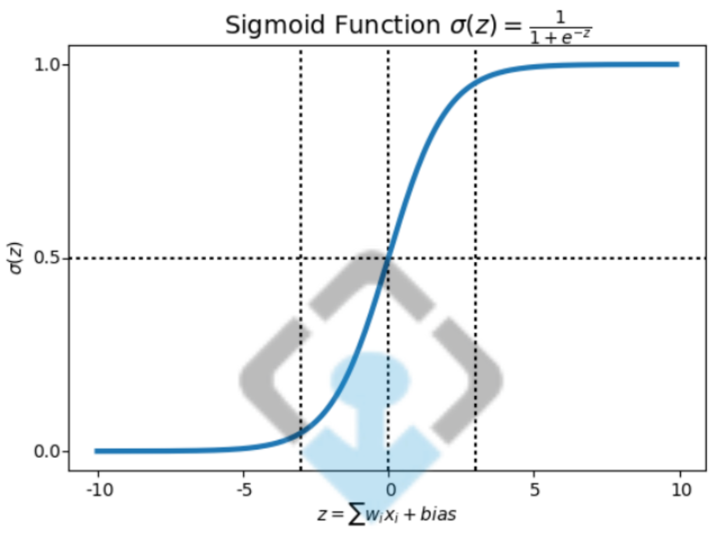
…..

Logistics Regression

18. What is Logistic Regression? What is Logistics Function or Sigmoid Function and What is the range of value of this function?

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The target or dependent variable can only have 2 possible classes (e.g.: Pass/Fail, 0/1, Spam/No-Spam etc.)

Logistics Function/Sigmoid Function: The Logistic Function also called the sigmoid function. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.



1 / (1 + e^-value)

The 'e' in the above equation represents the S-shaped curve that has values between 0 and 1. We write the equation for logistic regression as follows:

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

In the above equation, b0 and b1 are the two coefficients of the input x. We estimate these two coefficients using "maximum likelihood estimation". Below is the maximum log-likelihood cost function for Logistic regression.

$$J(\boldsymbol{w}) = -\sum_i y^{(i)} \log\left(\phi\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - \phi\left(z^{(i)}\right)\right)$$

To get a better grasp on this cost function, let's look at the cost that we calculate for one single-sample instance:

$$J(\phi(z), y; \mathbf{w}) = -y \log(\phi(z)) - (1-y)\log(1-\phi(z))$$

Looking at the preceding equation, we can see that the first term becomes zero if y = 0, and the second term becomes zero if y =1, respectively:

$$J(\phi(z), y; \mathbf{w}) = \begin{cases} -\log(\phi(z)) & if\ y = 1 \\ -\log(1-\phi(z)) & if\ y = 0 \end{cases}$$

19. Why is logistics regression despite being a classification algorithm have "Regression" in its name?

Although the dependent variable in logistic regression is binary, the predicted log odds can be any real number between negative infinity and positive infinity. In other words, your predictions are continuous. In order to use logistic regression for classification, you need to convert the log odds to a probability (which is also continuous from 0 to 1) and then set a probability threshold. The choice of threshold is done outside the model.

20. What is Maximum Likelihood?

It is a method in statistics for estimating parameter(s) of a model for given data. The basic intuition behind MLE is that the estimate which explains the data best, will be the best estimator.

The main advantage of MLE is that it has asymptotic property. It means that when the size of the data increases, the estimate converges faster towards the population parameter. We use MLE for many techniques in statistics to estimate parameters. I have explained the general steps we follow to find an estimate for the parameter.

Step 1: *Make an assumption about the data generating function.*

Step 2: *Formulate the likelihood function for the data, using the data generating function.*

The likelihood function is nothing but the probability of observing this data given the parameters (P(D|θ)). The parameters depend on our assumptions and the data-generating function.

Step 3: *Find an estimator for the parameter using an optimization technique.*

This is done by finding the estimate that maximizes the likelihood function. This is the reason why we name the estimator calculated using MLE as M-estimator.

Example: We have tossed a coin **n** times and observed **k** heads. Note that we consider the read is a success the and tail is a failure.

Step 1: *(Assumption):* The coin follows the Bernoulli distribution function.

Step 2: *(Likelihood Function):* The likelihood function is binomial distribution function P(D|θ)) in this case. We need to find out the best estimate for p (Probability of getting head) given that k of n tosses are Heads.
Step 3: *(Estimation):* M- estimator is

$$P = k/\text{n}$$

21. What is odds ratio?

If the probability of something happening is *p*, the odds-ratio is given by *p*/(1-*p*).

Example:
Suppose in a throw of a fair dice, A is the event that either 1 or 2 will surface.

So, p(A) = p = 1/3 and hence odds of A = (1/3)/(2/3) = 1/2

Odds of A is 50%, in this case, which can be explained as the likelihood of the event A is 50% of the likelihood of the complementary event of A.

22. Can we use Linear Regression in place of Logistics Regression for classification?

No. The reasons why linear regressions cannot be used in case of binary classification are as follows:
Distribution of error terms: The distribution of data in case of linear and logistic regression is different. Linear regression assumes that error terms are normally distributed. In case of binary classification, this assumption does not hold true.
Model output: In linear regression, the output is continuous. In case of binary classification, an output of a continuous value does not make sense. For binary classification problems, linear regression may predict values that can go beyond 0 and 1. If we want the output in the form of probabilities, which can be mapped to two different classes, then its range should be restricted to 0 and 1. As the logistic regression model can output probabilities with logistic/sigmoid function, it is preferred over linear regression.
Variance of Residual errors: Linear regression assumes that the variance of random errors is constant. This assumption is also violated in case of logistic regression.

23. What are the assumptions of Logistics Regression?

1. The logistic regression assumes that there is minimal or no multicollinearity among the independent variables.
2. The Logistic regression assumes that the independent variables are linearly related to the log of odds.
3. The logistic regression usually requires a large sample size to predict properly.
4. The Logistic regression which has two classes assumes that the dependent variable is binary and ordered logistic regression requires the dependent variable to be ordered.
5. The Logistic regression assumes the observations to be independent of each other.

24. What is the decision boundary in case of Logistics Regression?

Decision boundary helps to differentiate probabilities into positive class and negative class.

For Logistic Regression, we only have Linear Decision Boundary. The Boundary or line that separates the two classes are chosen by setting a threshold probability.

Step1 : Sigmoid gave the output in the range 0 to 1.

Step2: Set a threshold probability (assume 0.5)

Step3: Classification will happen by separating the points with probability less than or greater than 0.5

(In the above example, 0.5 is a linear decision boundary).

25. Which cost function is used in Logistics Regression? Why cannot we use MSE (Mean Squared Error) as a cost function in Logistics Regression?

The cost function used in Logistics Regression is Sigmoid Function or Logistic Function.
The main reason not to use the MSE as the cost function for logistic regression is because you don't want your cost function to be non-convex in nature. If the cost function is not convex then it is difficult for the function to optimally converge.

26. Can logistics regression handle categorical variables directly? If not, then how to use categorical value in Logistics Regression?

The inputs to a logistic regression model need to be numeric. The algorithm cannot handle categorical variables directly. So, they need to be converted into a format that is suitable for the algorithm to process.

The various levels of a categorical variable will be assigned a unique numeric value known as the dummy variable. These dummy variables are handled by the logistic regression model as any other numeric value.

27. How does Logistics Regression deals with Multiclass Classification? What is one-vs-all method?

The most famous method of dealing with multiclass classification using logistic regression is using the **one-vs-all approach**. Under this approach, a number of models are trained, which is equal to the number of classes. The models work in a specific way.

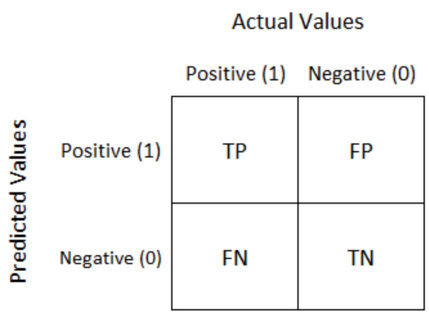
For example, the first model classifies the datapoint depending on whether it belongs to class 1 or some other class; the second model classifies the datapoint into class 2 or some other class. This way, each data point can be checked over all the classes.

28. What is the evaluation matrix for Classification Algorithms?

The different ways are as follows:

- Confusion matrix
- Accuracy
- Precision
- Recall
- Specificity
- F1 score
- Precision-Recall or PR curve
- ROC (Receiver Operating Characteristics) curve
- PR vs ROC curve

Confusion Matrix: it is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Definition of the terms:

Positive (P) : Observation is positive (for example: is an apple).
Negative (N) : Observation is not positive (for example: is not an apple).
True Positive (TP) : Observation is positive, and is predicted to be positive.
False Negative (FN) : Observation is positive, but is predicted negative.
True Negative (TN) : Observation is negative, and is predicted to be negative.
False Positive (FP) : Observation is negative, but is predicted positive.

Classification Rate/Accuracy: Classification Rate or Accuracy is given by the relation:

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

However, there are problems with accuracy. It assumes equal costs for both kinds of errors. A 99% accuracy can be excellent, good, mediocre, poor or terrible depending upon the problem.

Recall/Sensitivity/True Positive Rate: Recall can be defined as the ratio of the total number of correctly classified positive examples divide to the total number of positive examples. High Recall indicates the class is correctly recognized (a small number of FN).

$$\textbf{\textit{Recall}} = \frac{TP}{TP + FN}$$

Precision: To get the value of precision we divide the total number of correctly classified positive examples by the total number of predicted positive examples. High Precision indicates an example labelled as positive is indeed positive (a small number of FP).

$$\textit{Precision} = \frac{TP}{TP + FP}$$

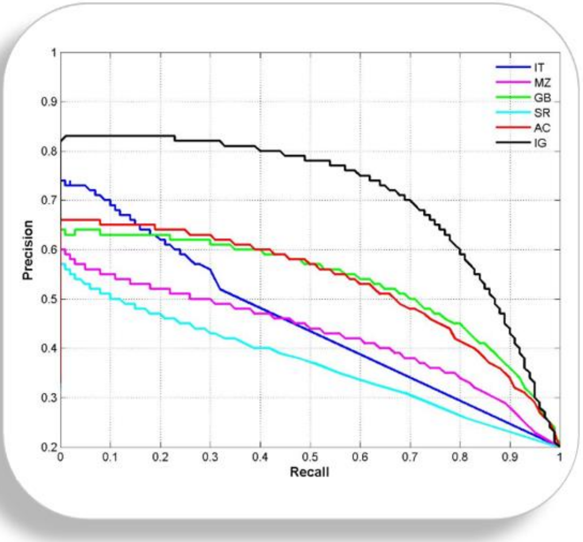
F-measure/F-stats/F1 Score: Since we have two measures (Precision and Recall) it helps to have a measurement that represents both of them. We calculate an F-measure which uses Harmonic Mean in place of Arithmetic Mean as it punishes the extreme values more.
The F-Measure will always be nearer to the smaller value of Precision or Recall.

$$\textit{F - measure} = \frac{2*Recall*Precision}{Recall + Precision}$$

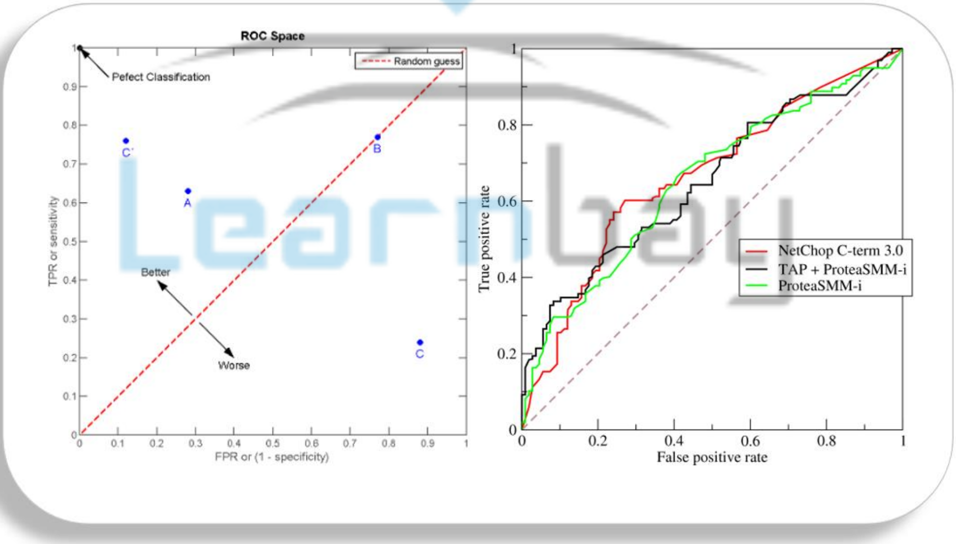
Specificity: Percentage of negative instances out of the total actual negative instances. Therefore, denominator (TN + FP) here is the actual number of negative instances present in the dataset. It is similar to recall but the shift is on the negative instances. Like finding out how many healthy patients were not having cancer and were told they don't have cancer. Kind of a measure to see how separate the classes are.

$$\frac{TN}{TN + FP}$$

PR Curve: It is the curve between precision and recall for various threshold values. In the figure below we have 6 predictors showing their respective precision-recall curve for various threshold values. The top right part of the graph is the ideal space where we get high precision and recall. Based on our application we can choose the predictor and the threshold value. PR AUC is just the area under the curve. The higher its numerical value the better.



ROC Curve: ROC stands for receiver operating characteristic and the graph is plotted against TPR and FPR for various threshold values. As TPR increases FPR also increases. As you can see in the first figure, we have four categories and we want the threshold value that leads us closer to the top left corner. Comparing different predictors (here 3) on a given dataset also becomes easy as you can see in figure 2, one can choose the threshold according to the application at hand. ROC AUC is just the area under the curve, the higher its numerical value the better.



PR vs ROC Curve:
Both the metrics are widely used to judge a model's performance.

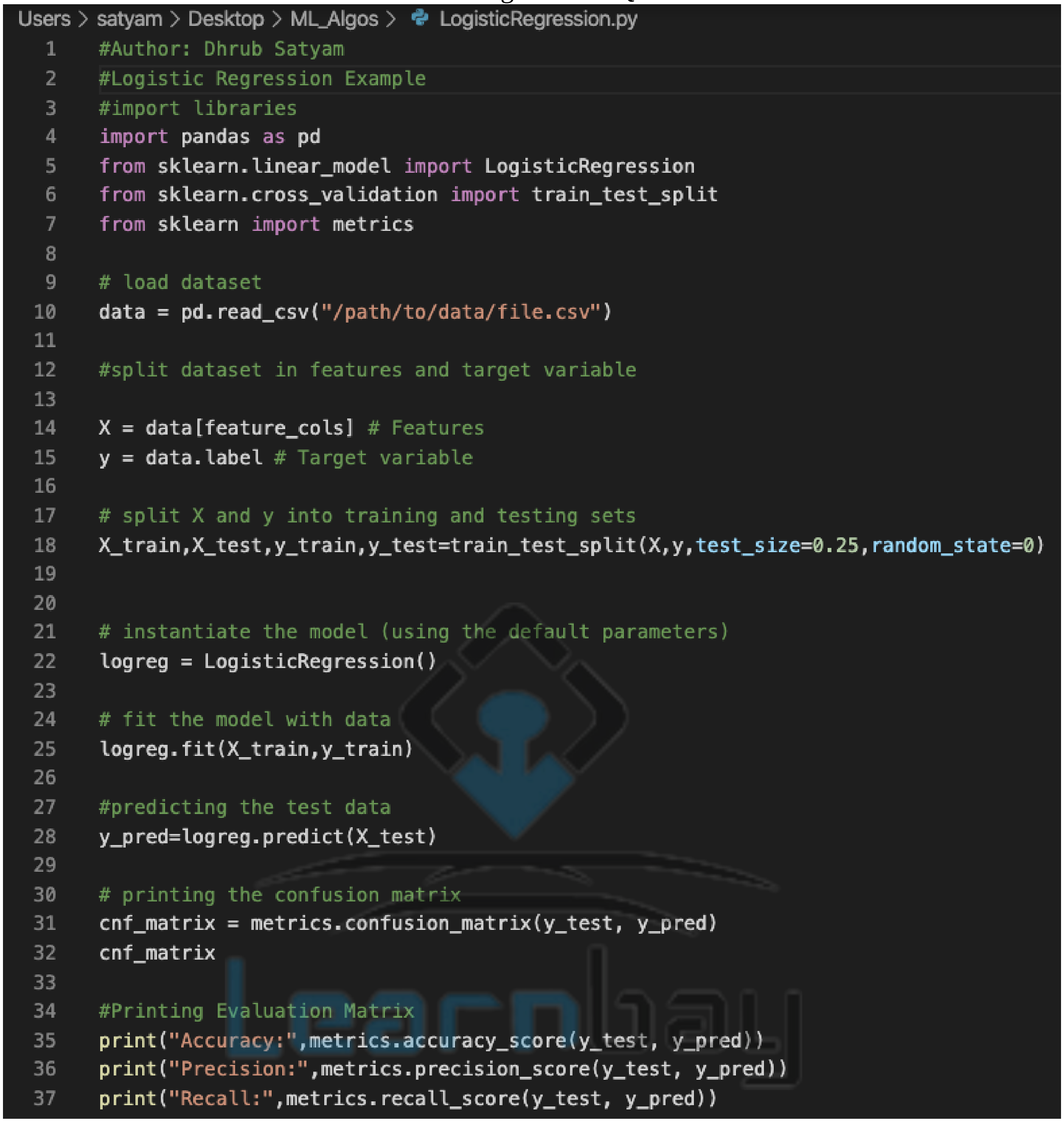
Which one to use PR or ROC?

The answer lies in TRUE NEGATIVES.

Due to the absence of TN in the precision-recall equation, they are useful in imbalanced classes. In the case of class imbalance when there is a majority of the negative class. The metric doesn't take much into consideration the high number of TRUE NEGATIVES of the negative class, which is in majority, giving better resistance to the imbalance. This is important when the detection of the positive class is very important.

Like to detect cancer patients, which has a high-class imbalance because very few have it out of all the diagnosed. We certainly don't want to miss on a person having cancer and going undetected (recall) and be sure the detected one is having it (precision).
Due to the consideration of TN or the negative class in the ROC equation, it is useful when both the classes are important to us. Like the detection of cats and dog. The importance of true negatives makes sure that both the classes are given importance, like the output of a CNN model in determining the image is of a cat or a dog.

29. Logistic Regression Code Snippet:



Users > satyam > Desktop > ML_Algos > LogisticRegression.py

```python
#Author: Dhrub Satyam
#Logistic Regression Example
#import libraries
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
from sklearn import metrics

# load dataset
data = pd.read_csv("/path/to/data/file.csv")

#split dataset in features and target variable

X = data[feature_cols] # Features
y = data.label # Target variable

# split X and y into training and testing sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)


# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X_train,y_train)

#predicting the test data
y_pred=logreg.predict(X_test)

# printing the confusion matrix
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix

#Printing Evaluation Matrix
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

.....

30. What is Decision Tree Algorithm?

Decision tree is a supervised machine learning algorithm mainly used for the **Regression and Classification**. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. Decision tree can handle both categorical and numerical data.

31. What are the different algorithms used in Decision Tree? Discuss.

Algorithms used in Decision Trees:

1. CART (Classification and Regression Trees): Uses Gini as splitting criteria.
2. C4.5 (successor of ID3): Uses Entropy as splitting Criteria.
3. ID3 (Iterative Dichotomiser 3): Uses Entropy as Splitting Criteria.
4. CHAID (Chi-square automatic interaction detection): Uses Chi-Square as splitting criteria.

32. What is Pruning? What are different types of Pruning?

Pruning is a technique used in Decision Tree that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

There are generally two methods for pruning trees: pre-pruning and post-pruning.

Pre-pruning is going to involve techniques that perform early stopping (e.g. we stop the building of our tree before it is fully grown).

Post-pruning will involve fully growing the tree in its entirety, and then trimming the nodes of the tree in a bottom-up fashion.

33. What is bagging?

Bagging, also called Bootstrap aggregating, is a machine learning ensemble algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach.

Let’s assume we have a sample dataset of 1000 instances (x) and we are using the CART algorithm. Bagging of the CART algorithm would work as follows.

- Create many (e.g. 100) random sub-samples of our dataset with replacement.
- Train a CART model on each sample.
- Given a new dataset, calculate the average prediction from each model.

34. What are advantages and disadvantages of using Decision Trees?

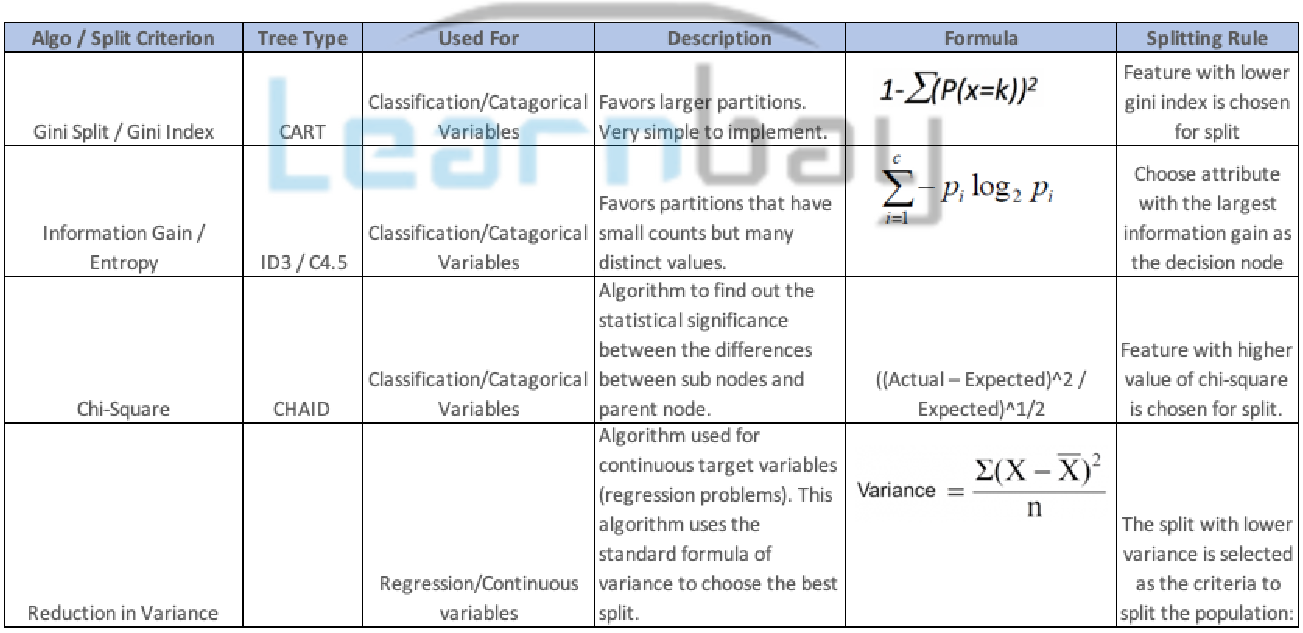
Advantages

1. Easy to Understand: Decision tree output is very easy to understand even for people from non-analytical background. It does not require any statistical knowledge to read and interpret them. Its graphical representation is very intuitive and users can easily relate their hypothesis.
2. Useful in Data exploration: Decision tree is one of the fastest way to identify most significant variables and relation between two or more variables. With the help of decision trees, we can create new variables / features that has better power to predict target variable. You can refer article (Trick to enhance power of regression model) for one such trick. It can also be used in data exploration stage. For example, we are working on a problem where we have information available in hundreds of variables, there decision tree will help to identify most significant variable.
3. Less data cleaning required: It requires less data cleaning compared to some other modelling techniques. It is not influenced by outliers and missing values to a fair degree.
4. Data type is not a constraint: It can handle both numerical and categorical variables.
5. Non-Parametric Method: Decision tree is considered to be a non-parametric method. This means that decision trees have no assumptions about the space distribution and the classifier structure.

Disadvantages

1. Over fitting: Over fitting is one of the most practical difficulty for decision tree models. This problem gets solved by setting constraints on model parameters and pruning (discussed in detailed below).
2. Not fit for continuous variables: While working with continuous numerical variables, decision tree loses information when it categorizes variables in different categories.
3. Decision Trees do not work well if you have smooth boundaries. i.e they work best when you have discontinuous piece wise constant model. If you truly have a linear target function decision trees are not the best.
4. Decision Tree's do not work best if you have a lot of un-correlated variables. Decision tree's work by finding the interactions between variables. if you have a situation where there are no interactions between variables linear approaches might be the best.
5. Data fragmentation: Each split in a tree leads to a reduced dataset under consideration. And, hence the model created at the split will potentially introduce bias.

35. What are the different splitting criteria used in Decision Tree? Discuss them in detail.

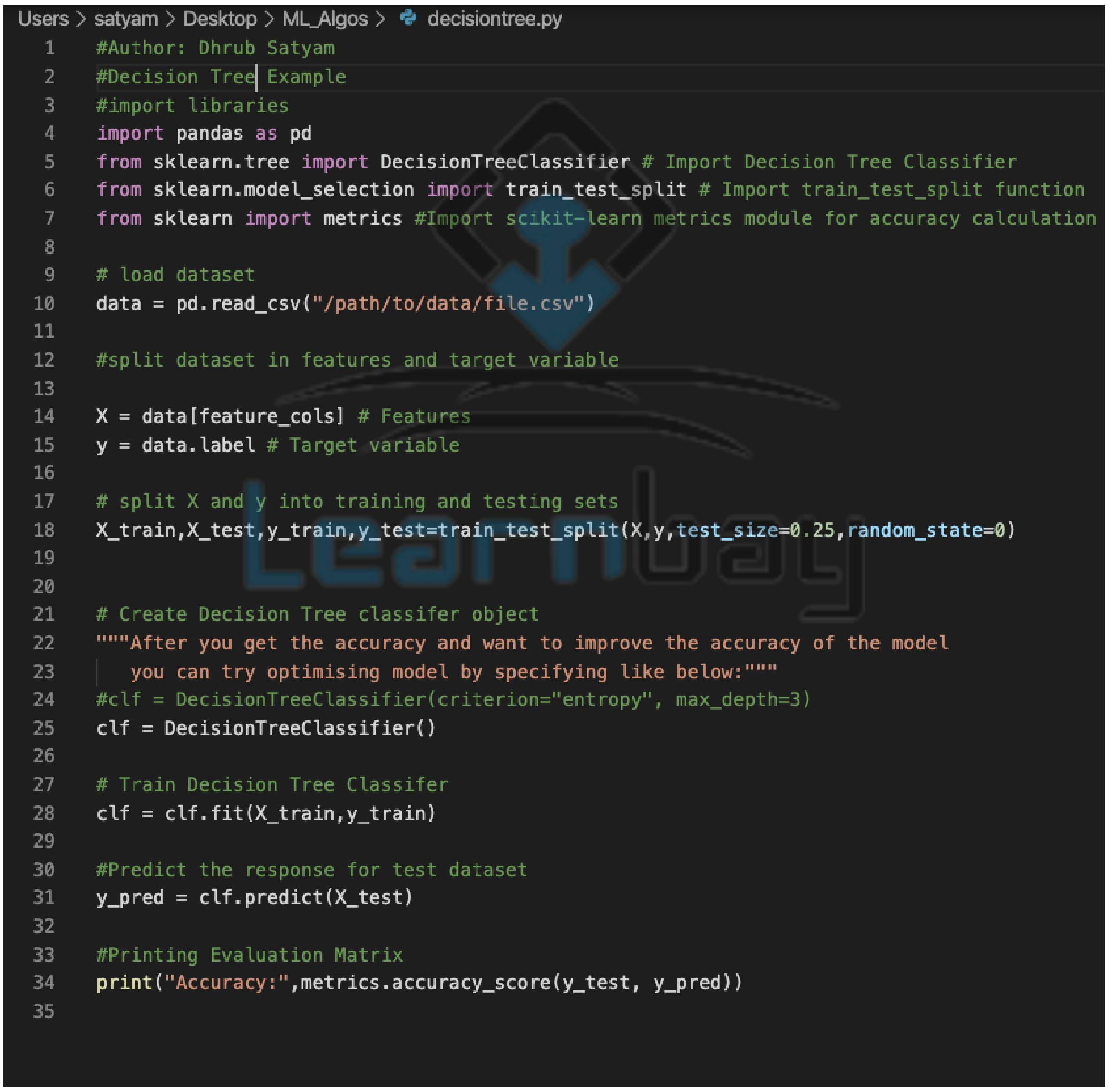


Algo / Split Criterion	Tree Type	Used For	Description	Formula	Splitting Rule
Gini Split / Gini Index	CART	Classification/Catagorical Variables	Favors larger partitions. Very simple to implement.	$1-\sum(P(x=k))^2$	Feature with lower gini index is chosen for split
Information Gain / Entropy	ID3 / C4.5	Classification/Catagorical Variables	Favors partitions that have small counts but many distinct values.	$\sum_{i=1}^{c} -p_i \log_2 p_i$	Choose attribute with the largest information gain as the decision node
Chi-Square	CHAID	Classification/Catagorical Variables	Algorithm to find out the statistical significance between the differences between sub nodes and parent node.	((Actual – Expected)^2 / Expected)^1/2	Feature with higher value of chi-square is chosen for split.
Reduction in Variance		Regression/Continuous variables	Algorithm used for continuous target variables (regression problems). This algorithm uses the standard formula of variance to choose the best split.	Variance $= \frac{\Sigma(X-\overline{X})^2}{n}$	The split with lower variance is selected as the criteria to split the population:

36. What is Ensemble Modelling? Why is it used?

Ensemble modelling is a process where multiple diverse models are created to predict an outcome, either by using many different modelling algorithms or using different training data sets. The ensemble model then aggregates the prediction of each base model and results in once final prediction for the unseen data. The motivation for using ensemble models is to reduce the generalization error of the prediction.
How to avoid Over-fitting in Decision Tree? Discuss the parameters/hyper parameters in detail.

37. Decision Tree Code Snippet:



Users > satyam > Desktop > ML_Algos > decisiontree.py

```python
#Author: Dhrub Satyam
#Decision Tree Example
#import libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

# load dataset
data = pd.read_csv("/path/to/data/file.csv")

#split dataset in features and target variable

X = data[feature_cols] # Features
y = data.label # Target variable

# split X and y into training and testing sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)


# Create Decision Tree classifer object
"""After you get the accuracy and want to improve the accuracy of the model
   you can try optimising model by specifying like below:"""
#clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

#Printing Evaluation Matrix
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

```

.....

38. What is Random Forest? Why is it preferred over Decision Trees?

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

39. What are the advantages and disadvantages of Random Forest?

Advantage

The following are the advantages of Random Forest algorithm −

- It overcomes the problem of overfitting by averaging or combining the results of different decision trees.
- Random forests work well for a large range of data items than a single decision tree does.
- Random forest has less variance then single decision tree.
- Random forests are very flexible and possess very high accuracy.
- Scaling of data does not require in random forest algorithm. It maintains good accuracy even after providing data without scaling.
- Random Forest algorithms maintains good accuracy even a large proportion of the data is missing.

Disadvantage

The following are the disadvantages of Random Forest algorithm −

- Complexity is the main disadvantage of Random forest algorithms.

- Construction of Random forests are much harder and time-consuming than decision trees.
- More computational resources are required to implement Random Forest algorithm.
- It is less intuitive in case when we have a large collection of decision trees.
- The prediction process using random forests is very time-consuming in comparison with other algorithms.

40. What are the different steps involved in Random Forest Modelling?

A random forest can be considered as an ensemble of decision trees. The idea behind ensemble learning is to combine weak learners to build a more robust model, a strong learner, that has a better generalization error and is less susceptible to overfitting.

The random forest algorithm can be summarized in four simple steps:
1. Draw a random bootstrap sample of size n (randomly choose n samples from the training set with replacement).
2. Grow a decision tree from the bootstrap sample. At each node:
   - Randomly select d features without replacement.
   - Split the node using the feature that provides the best split according to the objective function, for instance, by maximizing the information gain.
3. Repeat the steps 1 to 2 k times.
4. Aggregate the prediction by each tree to assign the class label by majority vote.

There is a slight modification in step 2 when we are training the individual decision trees: instead of evaluating all features to determine the best split at each node, we only consider a random subset of those.

41. What is cross validation? Why is it used?

The technique of cross validation (CV) is best explained by example using the most common method, K-Fold CV When we approach a machine learning problem, we make sure to split our data into a training and a testing set.

In K-Fold CV, we further split our training set into K number of subsets, called folds. We then iteratively fit the model K times, each time training the data on K-1 of the folds and evaluating on the Kth fold (called the validation data).

As an example, consider fitting a model with K = 5. The first iteration we train on the first four folds and evaluate on the fifth. The second time we train on the first, second, third, and fifth fold and evaluate on the fourth. We repeat this procedure 3 more times, each time evaluating on a different fold. At the very end of training, we average the performance on each of the folds to come up with final validation metrics for the model.

For hyperparameter tuning, we perform many iterations of the entire K-Fold CV process, each time using different model settings. We then compare all of the models, select the best one, train it on the full training set, and then evaluate on the testing set. If we have 10 sets of hyperparameters and are using 5-Fold CV, that represents 50 training loops.

Fortunately, as with most problems in machine learning, someone has solved our problem and model tuning with K-Fold CV can be automatically implemented in Scikit-Learn.

42. What is OOB (Out Of Bag) Error in Random Forest?

Out-of-bag (OOB) error, also called out-of-bag estimate, is a method of measuring the prediction error of random forests, boosted decision trees, and other machine learning models utilizing bootstrap aggregating (bagging) to sub-sample data samples used for training.

Subsampling allows one to define an out-of-bag estimate of the prediction performance improvement by evaluating predictions on those observations which were not used in the building of the next base learner.

In Layman Term: When you train each tree in random forest, you will not use all the samples. So for each bag, those unused samples can be used to find the prediction error for that particular bag. The OOB error rate can then be obtained by averaging the prediction error from all the bags.

43. What are different Hyperparameter tuning involved in Random Forest?

The different hyperparameters involved in Random Forest are:

1. max_depth : The max_depth of a tree in Random Forest is defined as the longest path between the root node and the leaf node. Using the max_depth parameter, I can limit up to what depth I want every tree in my random forest to grow.

2. min_sample_split: a parameter that tells the decision tree in a random forest the minimum required number of observations in any given node in order to split it.

3. max_leaf_nodes: This hyperparameter sets a condition on the splitting of the nodes in the tree and hence restricts the growth of the tree. The condition is based on the maximum number of leaf nodes allowed.

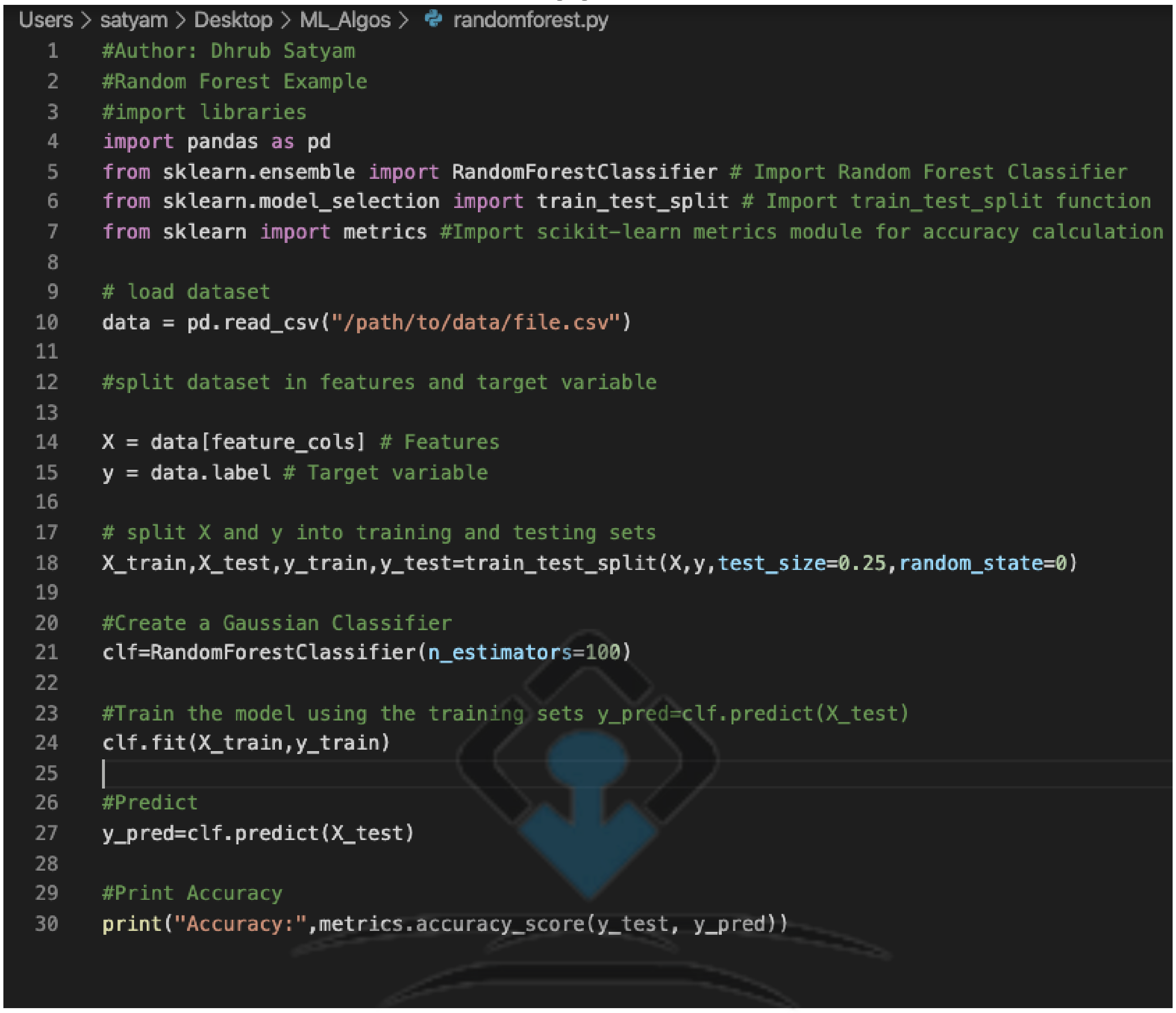
4. min_samples_leaf : This Random Forest hyperparameter specifies the minimum number of samples that should be present in the leaf node after splitting a node.

5. n_estimators: This is the number of trees you want to build before taking the maximum voting or averages of predictions. Higher number of trees give you better performance but makes your code slower.

6. max_sample (bootstrap sample): The max_samples hyperparameter determines what fraction of the original dataset is given to any individual tree.

7. max_features : This resembles the number of maximum features provided to each tree in a random forest.

44. Random Forest Code Snippet:



Users > satyam > Desktop > ML_Algos > randomforest.py

```python
#Author: Dhrub Satyam
#Random Forest Example
#import libraries
import pandas as pd
from sklearn.ensemble import RandomForestClassifier # Import Random Forest Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

# load dataset
data = pd.read_csv("/path/to/data/file.csv")

#split dataset in features and target variable

X = data[feature_cols] # Features
y = data.label # Target variable

# split X and y into training and testing sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

#Predict
y_pred=clf.predict(X_test)

#Print Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

.....

45. What is KNN Algorithm? How does it work?

The K Nearest Neighbors algorithm is a classification algorithm used in Data Science and Machine Learning.

The goal is to classify a new data point/observation into one of multiple existing categories. So, a number of neighbors ‘k’ is selected (usually k = 5), and the k closest data points are identified (either using Euclidean or Manhattan distances)

Of the k closest data points (or ‘neighbors’), the category with the highest number of k-close neighbors is the category assigned to the new data point.

Intuitively, this makes sense - a data point belongs in the category it's most similar to with respect to its features/properties. The most similar data points are the ones that are nearest to that data point, if you visualize it on a graph.

1. On a new data point, a kNN will calculate it's distance from *every single data point* in our dataset. The most popular distance metric used is the **Euclidean Distance.**
2. Once every single distance is calculated, the algorithm will pick the **K** nearest data points.
3. For classification problems, it'll make a prediction based on the class of those k-nearest datapoints. In this context, it's a good idea to choose K as an odd number to avoid dies. The predicted class will be the most frequent occuring class within K data points. For regression, it can predict based on the mean or median of those data points.

46. What is "K" in KNN Algorithm?

K = Number of nearest neighbours you want to select to predict the class of a given item

47. How to decide the value of K in KNN Algorithm? Why is odd value of K preferable while choosing?

Short Answer:

If K is small, then results might not be reliable because noise will have a higher influence on the result. If K is large, then there will be a lot of processing which may adversely impact the performance of the algorithm. So, following is must be considered while choosing the value of K:

a. K should be the square root of n (number of data points in training dataset)
b. K should be odd so that there are no ties. If square root is even, then add or subtract 1 to it.

K should be odd so that there are no ties in the voting. If square root of number of data points is even, then add or subtract 1 to it to make it odd.

Long Answer:

There is no straightforward method to calculate the value of K in KNN. You have to play around with different values to choose the optimal value of K. Choosing a right value of K is a process called Hyperparameter Tuning.

The value of optimum K totally depends on the dataset that you are using. The best value of K for KNN is highly data-dependent. In different scenarios, the optimum K may vary. It is more or less hit and trail method.

You need to maintain a balance while choosing the value of K in KNN. K should not be too small or too large.

A small value of K means that noise will have a higher influence on the result.

Larger the value of K, higher is the accuracy. If K is too large, you are under-fitting your model. In this case, the error will go up again. So, at the same time you also need to prevent your model from under-fitting. Your model should retain generalization capabilities otherwise there are fair chances that your model may perform well in the training data but drastically fail in the real data. Larger K will also increase the computational expense of the algorithm.

There is no one proper method of estimation of K value in KNN. No method is the rule of thumb but you should try considering following suggestions:

1. Square Root Method: Take square root of the number of samples in the training dataset.

2. Cross Validation Method: We should also use cross validation to find out the optimal value of K in KNN. Start with K=1, run cross validation (5 to 10 fold), measure the accuracy and keep repeating till the results become consistent.

K=1, 2, 3... As K increases, the error usually goes down, then stabilizes, and then raises again. Pick the optimum K at the beginning of the stable zone. This is also called Elbow Method.

3. Domain Knowledge also plays a vital role while choosing the optimum value of K.

4. K should be an odd number.

I would suggest to try a mix of all the above points to reach any conclusion.

48. Why is KNN Algorithm called a Lazy Learner? Can we use KNN for large datasets?

When it gets the training data, it does not learn and make a model, it just stores the data. It does not derive any discriminative function from the training data. It uses the training data when it actually needs to do some prediction. So, KNN does not immediately learn a model, but delays the learning, that is why it is called lazy learner.

KNN works well with smaller dataset because it is a lazy learner. It needs to store all the data and then makes decision only at run time. It needs to calculate the distance of a given point with all other points. So if dataset is large, there will be a lot of processing which may adversely impact the performance of the algorithm.

KNN is also very sensitive to noise in the dataset. If the dataset is large, there are chances of noise in the dataset which adversely affect the performance of KNN algorithm. For each new data point, the kNN classifier must:

1. Calculate the distances to all points in the training set and store them
2. Sort the calculated distances
3. Store the K nearest points
4. Calculate the proportions of each class
5. Assign the class with the highest proportion

Obviously, this is a very taxing process, both in terms of time and space complexity. The first operation is a quadratic time process, and the sorting

a O(n log n) process. Together, one could say that the process is a $O(n^3 \log n)$ process; a monstrously long process indeed.
Another problem is memory, since all pairwise distances must be stored and sorted in memory on a machine. With very large datasets, local machines will usually crash.

49. Discuss the advantages and disadvantages of KNN Algorithm.

Advantages of KNN

1. No Training Period: KNN is called Lazy Learner (Instance based learning). It does not learn anything in the training period. It does not derive any discriminative function from the training data. In other words, there is no training period for it. It stores the training dataset and learns from it only at the time of making real time predictions. This makes the KNN algorithm much faster than other algorithms that require training e.g. SVM, Linear Regression etc.

2. Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly which will not impact the accuracy of the algorithm.

3. KNN is very easy to implement. There are only two parameters required to implement KNN i.e. the value of K and the distance function (e.g. Euclidean or Manhattan etc.)

Disadvantages of KNN

1. Does not work well with large dataset: In large datasets, the cost of calculating the distance between the new point and each existing points is huge which degrades the performance of the algorithm.

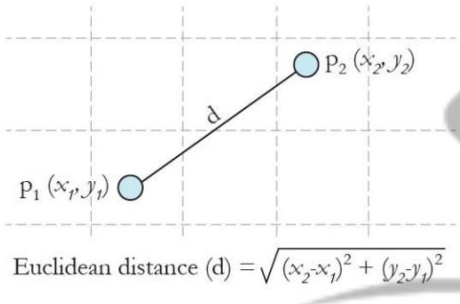
2. Does not work well with high dimensions: The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate the distance in each dimension.

3. Need feature scaling: We need to do feature scaling (standardization and normalization) before applying KNN algorithm to any dataset. If we don't do so, KNN may generate wrong predictions.

4. Sensitive to noisy data, missing values and outliers: KNN is sensitive to noise in the dataset. We need to manually impute missing values and remove outliers.

50. What are Euclidean, Manhattan and Chebyshev distance?

The Euclidean distance or Euclidean metric is the "ordinary" (i.e.straight-line) distance between two points in Euclidean space.



Euclidean distance (d) $= \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

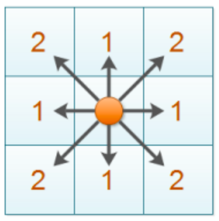
The Manhattan distance, also known as rectilinear distance, city block distance, taxicab metric is defined as the sum of the lengths of the projections of the line segment between the points onto the coordinate axes.
In chess, the distance between squares on the chessboard for rooks is measured in Manhattan distance.

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

distance between squares on the chessboard for **rooks** is measured in

ıce.

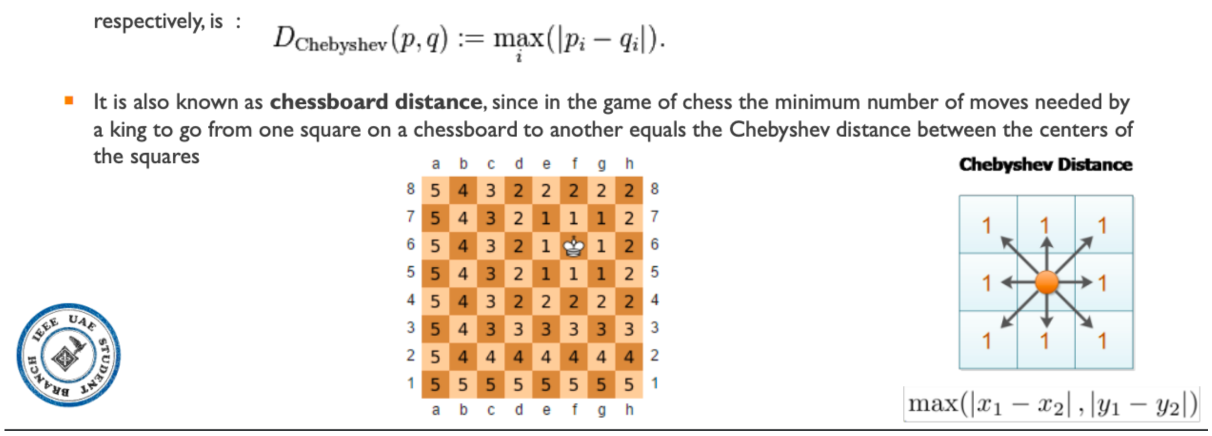
Manhattan Distance



$$|x_1 - x_2| + |y_1 - y_2|$$

The Chebyshev distance between two vectors or points p and q, with standard coordinates and respectively, is:

It is also known as chessboard distance, since in the game of chess the minimum number of moves needed by a king to go from one square on a chessboard to another equals the Chebyshev distance between the centres of the squares

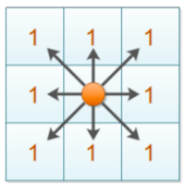


respectively, is : $D_{\text{Chebyshev}}(p,q) := \max_i(|p_i - q_i|).$

- It is also known as **chessboard distance**, since in the game of chess the minimum number of moves needed by a king to go from one square on a chessboard to another equals the Chebyshev distance between the centers of the squares

	a	b	c	d	e	f	g	h	
8	5	4	3	2	2	2	2	2	8
7	5	4	3	2	1	1	1	2	7
6	5	4	3	2	1	♔	1	2	6
5	5	4	3	2	1	1	1	2	5
4	5	4	3	2	2	2	2	2	4
3	5	4	3	3	3	3	3	3	3
2	5	4	4	4	4	4	4	4	2
1	5	5	5	5	5	5	5	5	1
	a	b	c	d	e	f	g	h	

Chebyshev Distance



$\max(|x_1 - x_2|, |y_1 - y_2|)$

Euclidean distance	Manhattan distance
It is computed as the hypotenuse like in the Pythagorean theorem.	It is computed as the sum of two sides of the right triangle but not the hypotenuse.
The mathematical equation to calculate Euclidean distance is : $Euclidean_{dis\tan ce} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ Where (x_1, y_1) and (x_2, y_2) are coordinates of the two points between whom the distance is to be determined.	The mathematical equation to calculate Manhattan distance is : $Manhat\tan_{dis\tan ce} = \lvert x_1 - x_2 \rvert + \lvert y_1 - y_2 \rvert$ Where (x_1, y_1) and (x_2, y_2) are coordinates of the two points between whom the distance is to be determined.
It the square roots of the sum of difference between the coordinates of the two points.	It the sum of absolute differences of their Cartesian coordinates.
If the points are very far then this will give wrong result because of the curvature of the earth.	It is more suitable natural settings and urban environment.
It always gives the shortest distance between the two points	It may give a longer distance between the two points

51. How to handle categorical variables in KNN?

Create dummy variables out of a categorical variable and include them instead of original categorical variable. Unlike regression, create k dummies instead of (k-1).

For example, a categorical variable named “Department” has 5 unique levels / categories. So we will create 5 dummy variables. Each dummy variable has 1 against its department and else 0.

52. Can KNN be used for Regression? How to use KNN for Regression?

Yes, K-nearest neighbour can be used for regression. In other words, K-nearest neighbour algorithm can be applied when dependent variable is continuous. In this case, the predicted value is the average of the values of its k nearest neighbours

53. Discuss the difference between KNN and K Means Algorithms.

KNN and k-means clustering both are very different algorithms that solve different problems and have their own meanings of what the variable ‘k’ is. KNN is a supervised classification algorithm that will label new data points based on the ‘k’ number of nearest data points and k-means clustering is an unsupervised clustering algorithm that groups the data into ‘k’ number of clusters.

54. How to reduce increased variance of the model other than changing k?

By using bagging-based decision boundary. If not restricted in the number of times, one can draw samples from the original dataset, a simple variance reduction method would be to sample, many times, and then simply take a majority vote of the kNN models fit to each of these samples to classify each test data point. This variance reduction method is called bagging.

55. What is the effect of sampling on KNN?

Sampling does several things in the perspective of a single data point, since kNN works on a point-by-point basis.

1. The average distance to the k nearest neighbours increases due to increased sparsity in the dataset.

2. Consequently, the area covered by k-nearest neighbours increases in size and covers a larger area of the feature space.
3. The sample variance increases.

A consequence to this change in input is an increase in variance. When we talk of variance, we refer to the variability in the predictions given different samples from the population. Why would the immediate effects of sampling lead to increased variance of the model?
Notice that now a larger area of the feature space is represented by the same k data points. While our sample size has not grown, the population space that it represents has increased in size. This will result in higher variance in the proportion of classes in the k nearest data points, and consequently a higher variance in the classification of each data point.

56. What happens when we change the value of K in KNN?

Short Answer: The class boundaries of the predictions become more smooth as k increases.

Long Answer: What really is the significance of these effects? First, it gives hints that a lower k value makes the kNN model more "sensitive." That is, it is more sensitive to the local changes in the dataset. The "sensitivity" of the model directly translates to its variance.

All of these examples point to an inverse relationship between variance and k. Additionally, consider how kNN operates when k reaches its maximum value, k=n, where n is the number of points in the training set) In this case, the majority class in the training set will always dominate the predictions. It will simply pick the most abundant class in the data, and never deviate, effectively resulting in zero variance. Therefore, it seems to reduce variance, k must be increased.

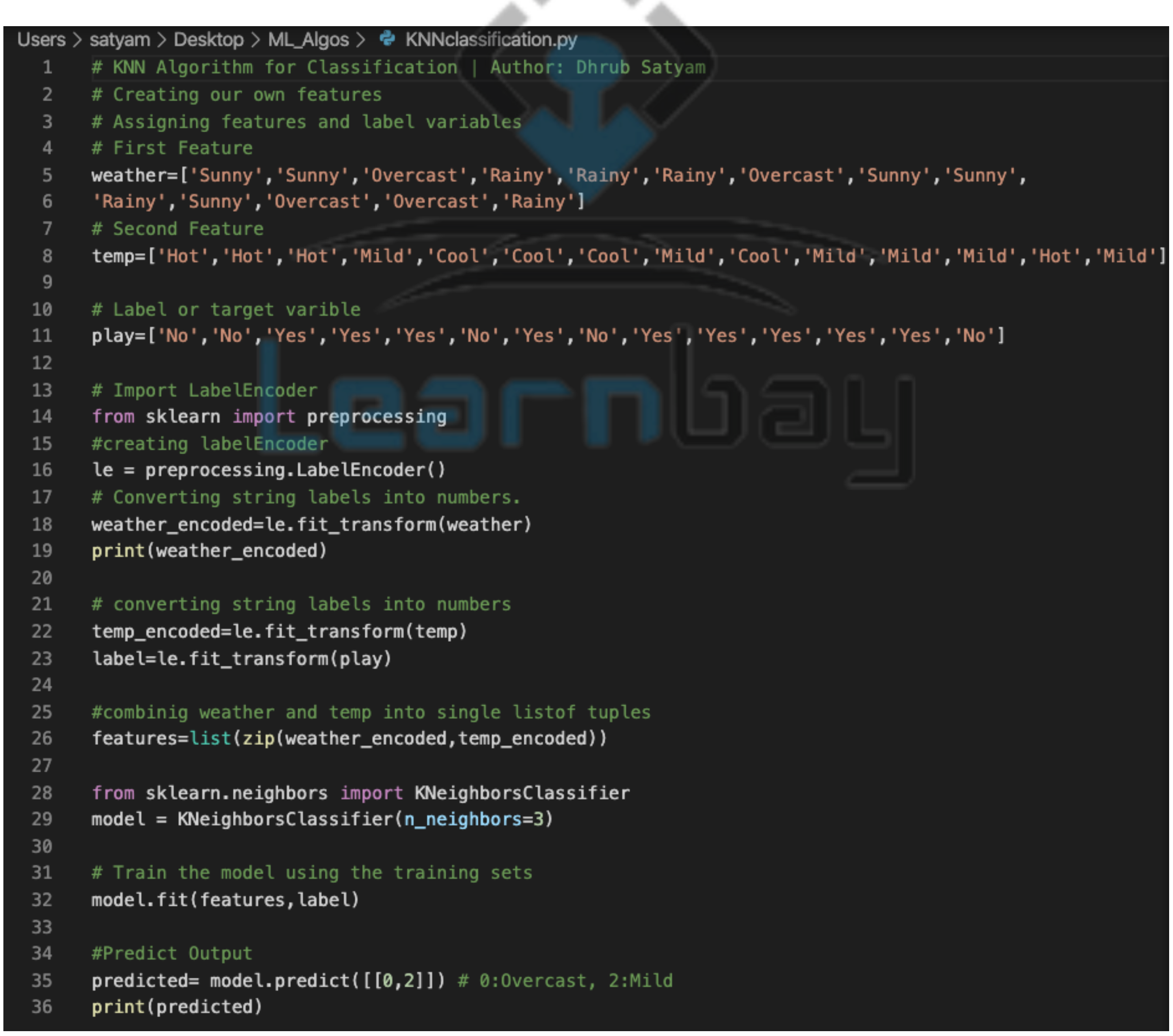
Final Verdict: In order to offset the increased variance due to sampling, k can be increased to decrease model variance.

57. What is the thumb rule to approach KNN problem?

1. Load the data
2. Initialise the value of k

3. For getting the predicted class, iterate from 1 to total number of training data points
    1. Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
    2. Sort the calculated distances in ascending order based on distance values
    3. Get top k rows from the sorted array
    4. Get the most frequent class of these rows
    5. Return the predicted class

58. KNN Code Snippet:

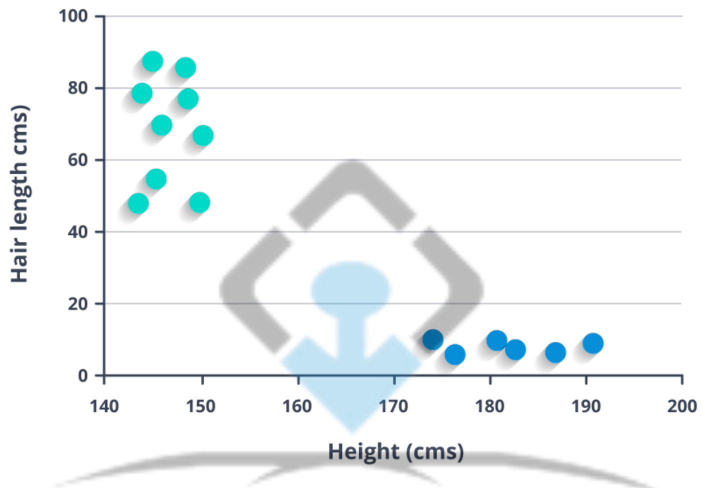


Users > satyam > Desktop > ML_Algos > KNNclassification.py

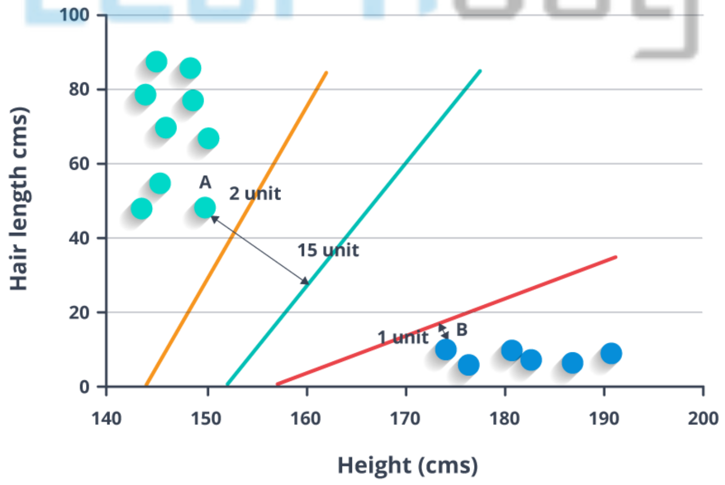
```python
# KNN Algorithm for Classification | Author: Dhrub Satyam
# Creating our own features
# Assigning features and label variables
# First Feature
weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny',
'Rainy','Sunny','Overcast','Overcast','Rainy']
# Second Feature
temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild']

# Label or target varible
play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']

# Import LabelEncoder
from sklearn import preprocessing
#creating labelEncoder
le = preprocessing.LabelEncoder()
# Converting string labels into numbers.
weather_encoded=le.fit_transform(weather)
print(weather_encoded)

# converting string labels into numbers
temp_encoded=le.fit_transform(temp)
label=le.fit_transform(play)

#combinig weather and temp into single listof tuples
features=list(zip(weather_encoded,temp_encoded))

from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)

# Train the model using the training sets
model.fit(features,label)

#Predict Output
predicted= model.predict([[0,2]]) # 0:Overcast, 2:Mild
print(predicted)
```

59. What is SVM Algorithm?

SVM stands for support vector machine, it is a supervised machine learning algorithm which can be used for both Regression and Classification.
In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.
For example, if we only had two features like Height and Hair length of an individual, we'd first plot these two variables in two-dimensional space where each point has two co-ordinates (these co-ordinates are known as Support Vectors)



Now, we will find some line that splits the data between the two differently classified groups of data. This will be the line such that the distances from the closest point in each of the two groups will be farthest away.



In the example shown above, the line which splits the data into two differently classified groups is the black line, since the two closest points are the farthest apart from the line. This line is our classifier. Then, depending on where the testing data lands on either side of the line, that's what class we can classify the new data as.

60. What are support Vectors?

A support vector machine attempts to find the line that "best" separates two classes of points. By "best", we mean the line that results in the largest margin between the two classes. The points that lie on this margin are the support vectors.
The vectors that define the hyperplane are the support vectors.

61. What is the purpose of the Support Vector in SVM?

A Support Vector Machine (SVM) performs classification by finding the hyperplane that maximizes the distance margin between the two classes.
The extreme points in the data sets that define the hyperplane are the support vectors

62. What are kernels?

SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types.

There are four types of kernels in SVM.

1. Linear Kernel
2. Polynomial kernel
3. Radial basis kernel
4. Sigmoid kernel

63. What is Kernel Trick?

Short Answer: It allows us to operate in the original feature space without computing the coordinates of the data in a higher dimensional space.

Long Answer:

1. For a dataset with n features (~n-dimensional), SVMs find an n-1-dimensional hyperplane to separate it (let us say for classification)
2. Thus, SVMs perform very badly with datasets that are not linearly separable
3. But, quite often, it's possible to transform our not-linearly-separable dataset into a higher-dimensional dataset where it becomes linearly separable, so that SVMs can do a good job
4. Unfortunately, quite often, the number of dimensions you have to add (via transformations) depends on the number of dimensions you already have (and not linearly)

a. For datasets with a lot of features, it becomes next to impossible to try out all the interesting transformations

5. Enter the Kernel Trick
    - Thankfully, the only thing SVMs need to do in the (higher-dimensional) feature space (while training) is computing the pair-wise dot products
    - For a given pair of vectors (in a lower-dimensional feature space) and a transformation into a higher-dimensional space, there exists a function (The Kernel Function) which can compute the dot product in the higher-dimensional space without explicitly transforming the vectors into the higher-dimensional space first
    - We are saved!
6. SVM can now do well with datasets that are not linearly separable

64. Why is SVM called as Large Margin Classifier?

Short Answer: Because it places the decision boundary such that it maximizes the distance between two clusters.

Long Answer: choosing the best hyperplane is to choose one in which the distance from the training points is the maximum. This is formalized by the geometric margin. Without getting into the details of the derivation, the geometric margin is given by:

$$\hat{\gamma}_i = \frac{\gamma_i}{||w||}.$$

Which is simply the functional margin normalized. So, these intuitions lead to the maximum margin classifier which is a precursor to the SVM.

65. What is the difference between Logistics Regression and SVM? When to use which model?

1. SVM tries to find the “best” margin (distance between the line and the support vectors) that separates the classes and this reduces the risk of error on the data, while logistic regression does not, instead it can have different decision boundaries with different weights that are near the optimal point.

2. SVM works well with unstructured and semi-structured data like text and images while logistic regression works with already identified independent variables.

3. SVM is based on the geometrical properties of the data while logistic regression is based on statistical approaches.

4. Logistic Regression can’t be applied to nonlinearly separable dataset whereas SVM can be applied.

5. The risk of overfitting is less in SVM, while Logistic regression is vulnerable to overfitting.

When to Use Logistic Regression vs Support Vector Machine?

Depending on the number of training sets (data)/features that you have, you can choose to use either logistic regression or support vector machine.

Let’s take these as an example where:

n = number of features,
m = number of training examples

1. If *n is large (1–10,000) and m is small (10–1000)*: use logistic regression or SVM with a linear kernel.
2. If *n is small (1–10 00) and m is intermediate (10–10,000)*: use SVM with (Gaussian, polynomial etc) kernel
3. If *n is small (1–10 00), m is large* (50,000–1,000,000+): first, manually add more features and then use logistic regression or SVM with a linear kernel

66. What does c and gamma parameter in SVM signify?

Short Answer:
Cost and Gamma are the hyper-parameters that decide the performance of an SVM model. There should be a fine balance between Variance and Bias for any ML model. (this is a science and an art - as we call it in empirical studies)
For SVM, a High value of Gamma leads to more accuracy but biased results and vice-versa. Similarly, a large value of Cost parameter (C) indicates poor accuracy but low bias and vice-versa.
Following table summarizes the above explanation -

The art is to choose a model with optimum variance and bias. Therefore, you need to choose the values of C and Gamma accordingly.
Optimum values of C and Gamma can be found by using methods like Gridsearch.

Long Answer:

The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'.

The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. If gamma is too large, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C will be able to prevent overfitting.

When gamma is very small, the model is too constrained and cannot capture the complexity or "shape" of the data. The region of influence of any selected support vector would include the whole training set. The resulting model will behave similarly to a linear model with a set of hyperplanes that separate the centres of high density of any pair of two classes.

67. What are Advantages and Disadvantages of SVM?

SVM Advantages

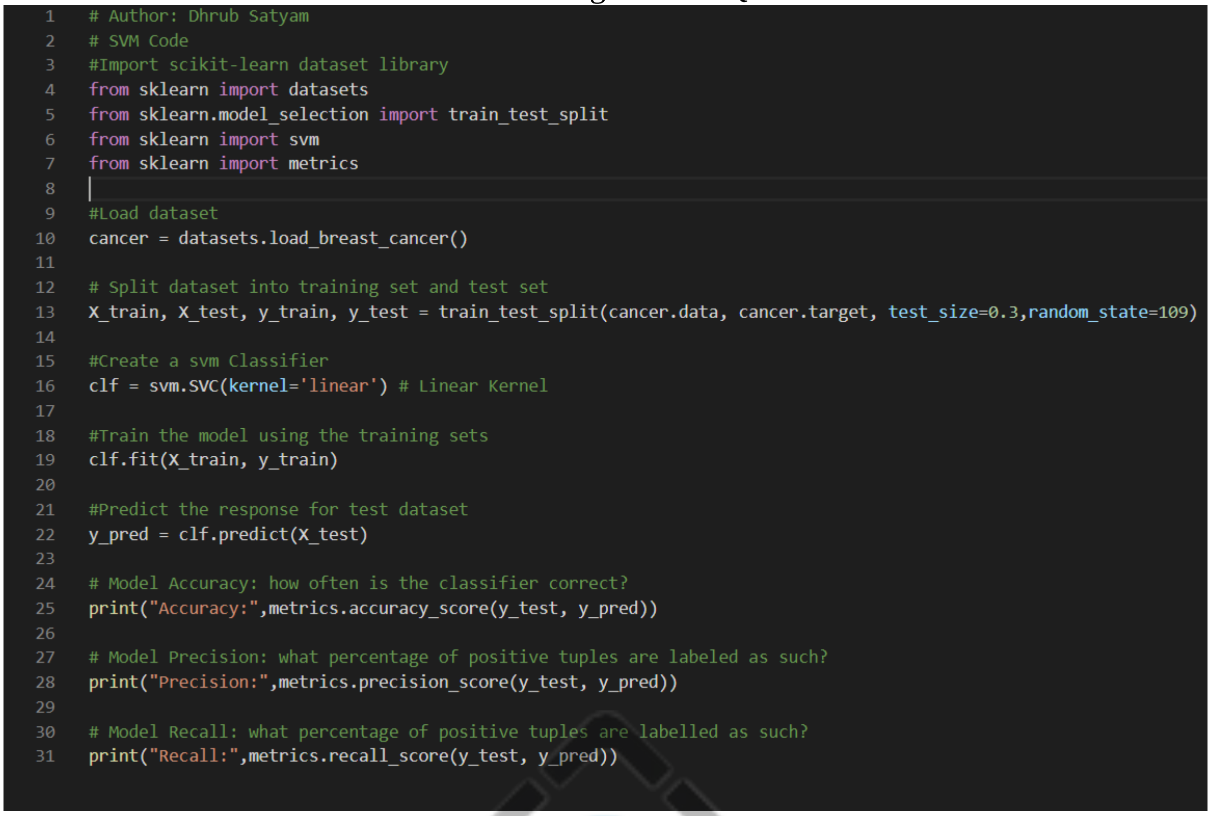
- SVM's are very good when we have no idea on the data.
- Works well with even unstructured and semi structured data like text, Images and trees.
- The kernel trick is real strength of SVM. With an appropriate kernel function, we can solve any complex problem.
- Unlike in neural networks, SVM is not solved for local optima.
- It scales relatively well to high dimensional data.
- SVM models have generalization in practice, the risk of over-fitting is less in SVM.
- SVM is always compared with ANN. When compared to ANN models, SVMs give better results.

SVM Disadvantages

- Choosing a "good" kernel function is not easy.
- Long training time for large datasets.

- Difficult to understand and interpret the final model, variable weights and individual impact.
- Since the final model is not so easy to see, we can not do small calibrations to the model hence its tough to incorporate our business logic.
- The SVM hyper parameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact.

68. SVM Code Snippet:



```python
# Author: Dhrub Satyam
# SVM Code
#Import scikit-learn dataset library
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics

#Load dataset
cancer = datasets.load_breast_cancer()

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3,random_state=109)

#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred))

# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred))
```

.....

Naïve Bayes

69. What is Naïve Bayes Algorithm?

It is a classification algorithm that predicts the probability of each data point belonging to a class and then classifies the point as the class with the highest probability.

70. Discuss Bayes Theorem.

Bayes' Theorem gives us the probability of an event actually happening by combining the conditional probability given some result and the prior knowledge of an event happening.

Conditional probability is the probability that something will happen, given that something has a occurred. In other words, the conditional probability is the probability of X given a test result or P(X|Test). For example, what is the probability an e-mail is spam given that my spam filter classified it as spam.

The prior probability is based on previous experience or the percentage of previous samples. For example, what is the probability that any email is spam. Formally

- P(A|B) = Posterior probability = Probability of A given B happened
- P(B|A) = Conditional probability = Probability of B happening if A is true
- P(A) = Prior probability = Probability of A happening in general
- P(B) = Evidence probability = Probability of getting a positive test

71. Why is Naïve Bayes Naïve?

In Layman's Term: The simple meaning of Naive is willing to believe that that life is simple and fair, which is not true. Naive Bayes is naive because it assumes that the features that are going into the model are not related to each other anyhow Change in one variable will not affect the other variable directly.

Long Answer: Naive Bayes (NB) is 'naive' because it makes the assumption that features of a measurement are independent of each other. This is naive because it is (almost) never true. Here is how it work even then - NB is a very intuitive classification algorithm. It asks the question, "Given these features, does this measurement belong to class A or B?", and answers it by taking the proportion of all previous measurements with the same features belonging to class A multiplied by the proportion of all measurements in class A. If this number is bigger than the corresponding calculation for class B then we say the measurement belongs in class A.

72. What are feature matrix and response vectors?

- Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of dependent features.
- Response vector contains the value of class variable (prediction or output) for each row of feature matrix.

73. Applications of Naïve Bayes Classification Algorithms?

Some of real-world examples are as given below

- To mark an email as spam, or not spam?
- Classify a news article about technology, politics, or sports?
- Check a piece of text expressing positive emotions, or negative emotions?
- Also used for face recognition software.

74. What are the Advantages and Disadvantages of using Naïve Bayes Algorithm?

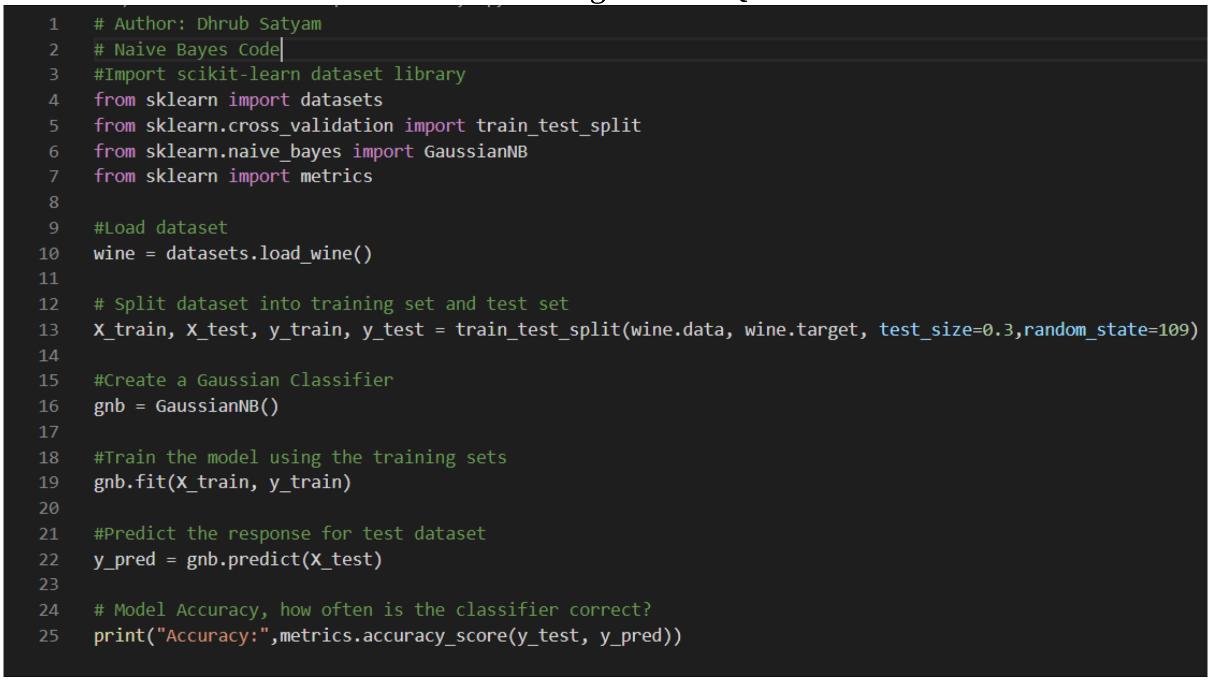
ADVANTAGES

1. Fast
2. Highly scalable.
3. Used for binary and Multi class Classification.
4. Great Choice for text classification.
5. Can easily train smaller data sets.

DISADVANTAGES

Naive Bayes considers that the features are independent of each other. However, in real world, features depend on each other.

75. Naïve Bayes Code Snippet:



```python
# Author: Dhrub Satyam
# Naive Bayes Code
#Import scikit-learn dataset library
from sklearn import datasets
from sklearn.cross_validation import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

#Load dataset
wine = datasets.load_wine()

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, test_size=0.3,random_state=109)

#Create a Gaussian Classifier
gnb = GaussianNB()

#Train the model using the training sets
gnb.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = gnb.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

76. What is K-Means Clustering? What are the steps for it?

K-means (Macqueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining.

If k is given, the K-means algorithm can be executed in the following steps:

- Partition of objects into k non-empty subsets
- Identifying the cluster centroids (mean point) of the current partition.
- Assigning each point to a specific cluster
- Compute the distances from each point and allot points to the cluster where the distance from the centroid is minimum.
- After re-allotting the points, find the centroid of the new cluster formed.

77. Why is the word "means" associated with the name of K-Means algorithm?

The ‘means’ in the K-means refers to **averaging of the data**; that is, finding the centroid.

There are k-medoids and k-medians algorithms as well.

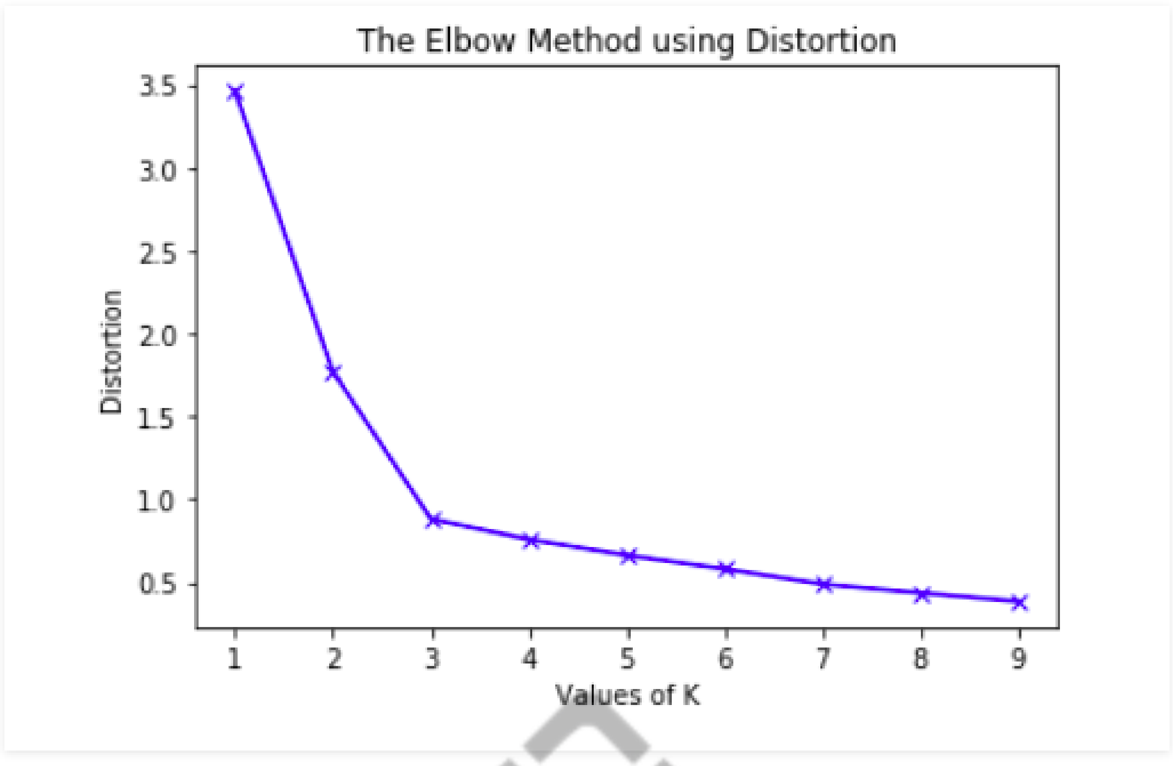
k-medoids minimizes the sum of dissimilarities between points labelled to be in a cluster and a point designated as the centre of that cluster. In contrast to the k-means algorithm, k-medoids chooses datapoints as centres (medoids or exemplars).

k-medians is a variation of k-means clustering where instead of calculating the mean for each cluster to determine its centroid, one instead calculates the median.

78. How to find the optimum number of clusters in K-Means? Discuss the elbow curve/elbow method?

Basic idea behind partitioning methods, such as k-means clustering, is to define clusters such that the total intra-cluster variation [or total within-cluster sum of square (WSS)] is minimized. The total WSS measures the compactness of the clustering and we want it to be as small as possible.

The Elbow method looks at the total WSS as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn’t improve much better the total WSS.



Notice the elbow at k =3.

The optimal number of clusters can be defined as follow:

1. Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, by varying k from 1 to 10 clusters.
2. For each k, calculate the total within-cluster sum of square (WSS).
3. Plot the curve of WSS according to the number of clusters k.
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

79. What is the difference between K-Means and Hierarchical Clustering? When to use which?

Hierarchical Clustering and k-means clustering complement each other. In hierarchical clustering, the researcher is not aware of the number of clusters to be made whereas in k-means clustering, the number of clusters to be made are specified before-hand.
Advice- If unaware about the number of clusters to be formed, use hierarchical

clustering to determine the number and then use k-means clustering to make more stable clusters as hierarchical clustering is a single-pass exercise whereas k-means is an iterative process.

80. What are the advantages and disadvantages of using K-Means Algorithms?

K-Means Advantages :

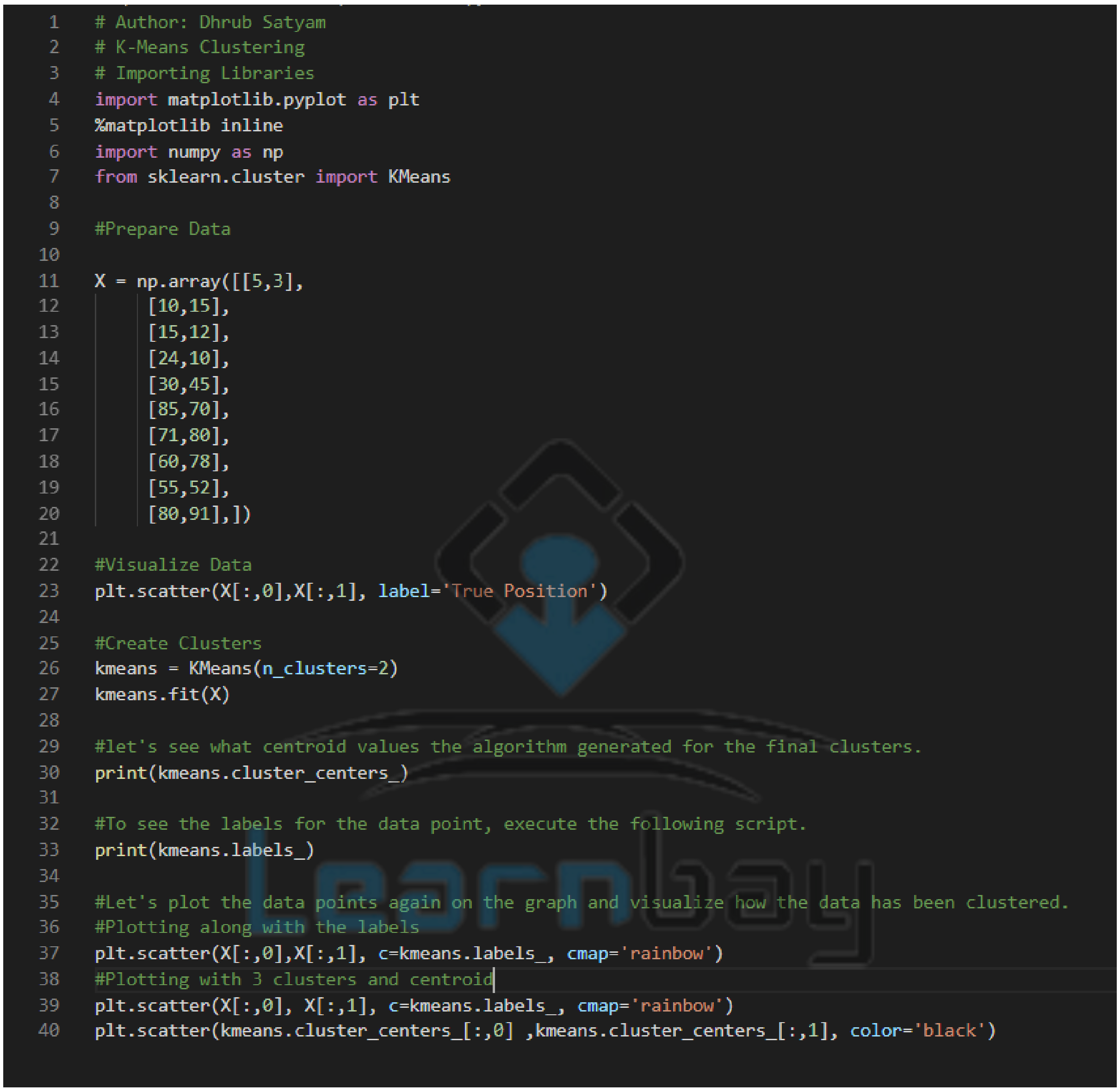
1) If variables are huge, then K-Means most of the times computationally faster than hierarchical clustering, if we keep k smalls.

2) K-Means produce tighter clusters than hierarchical clustering, especially if the clusters are globular.

K-Means Disadvantages :

1) Difficult to predict K-Value.
2) With global cluster, it didn't work well.
3) Different initial partitions can result in different final clusters.
4) It does not work well with clusters (in the original data) of Different size and Different density

81. K-Means Code Snippet:



```
# Author: Dhrub Satyam
# K-Means Clustering
# Importing Libraries
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from sklearn.cluster import KMeans

#Prepare Data

X = np.array([[5,3],
        [10,15],
        [15,12],
        [24,10],
        [30,45],
        [85,70],
        [71,80],
        [60,78],
        [55,52],
        [80,91],])

#Visualize Data
plt.scatter(X[:,0],X[:,1], label='True Position')

#Create Clusters
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

#let's see what centroid values the algorithm generated for the final clusters.
print(kmeans.cluster_centers_)

#To see the labels for the data point, execute the following script.
print(kmeans.labels_)

#Let's plot the data points again on the graph and visualize how the data has been clustered.
#Plotting along with the labels
plt.scatter(X[:,0],X[:,1], c=kmeans.labels_, cmap='rainbow')
#Plotting with 3 clusters and centroid
plt.scatter(X[:,0], X[:,1], c=kmeans.labels_, cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[:,0] ,kmeans.cluster_centers_[:,1], color='black')
```

82. What is Hierarchical Clustering?

Hierarchical clustering is another unsupervised learning algorithm that is used to group together the unlabelled data points having similar characteristics. Hierarchical clustering algorithms falls into following two categories.

Agglomerative hierarchical algorithms – In agglomerative hierarchical algorithms, each data point is treated as a single cluster and then successively merge or agglomerate (bottom-up approach) the pairs of clusters. The hierarchy of the clusters is represented as a dendrogram or tree structure.

Divisive hierarchical algorithms − On the other hand, in divisive hierarchical algorithms, all the data points are treated as one big cluster and the process of clustering involves dividing (Top-down approach) the one big cluster into various small clusters.

83. What are the steps to perform Agglomerative Hierarchical Clustering?

Most used and important Hierarchical clustering i.e. agglomerative. The steps to perform the same is as follows −

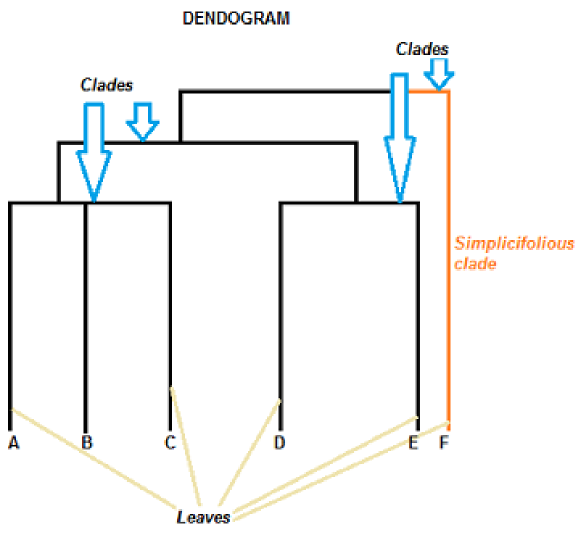
- **Step 1** − Treat each data point as single cluster. Hence, we will be having, say K clusters at start. The number of data points will also be K at start.
- **Step 2** − Now, in this step we need to form a big cluster by joining two closet datapoints. This will result in total of K-1 clusters.
- **Step 3** − Now, to form more clusters we need to join two closet clusters. This will result in total of K-2 clusters.
- **Step 4** − Now, to form one big cluster repeat the above three steps until K would become 0 i.e. no more data points left to join.
- **Step 5** − At last, after making one single big cluster, dendrograms will be used to divide into multiple clusters depending upon the problem.

84. What is Dendogram and what is its importance in Hierarchical Clustering?

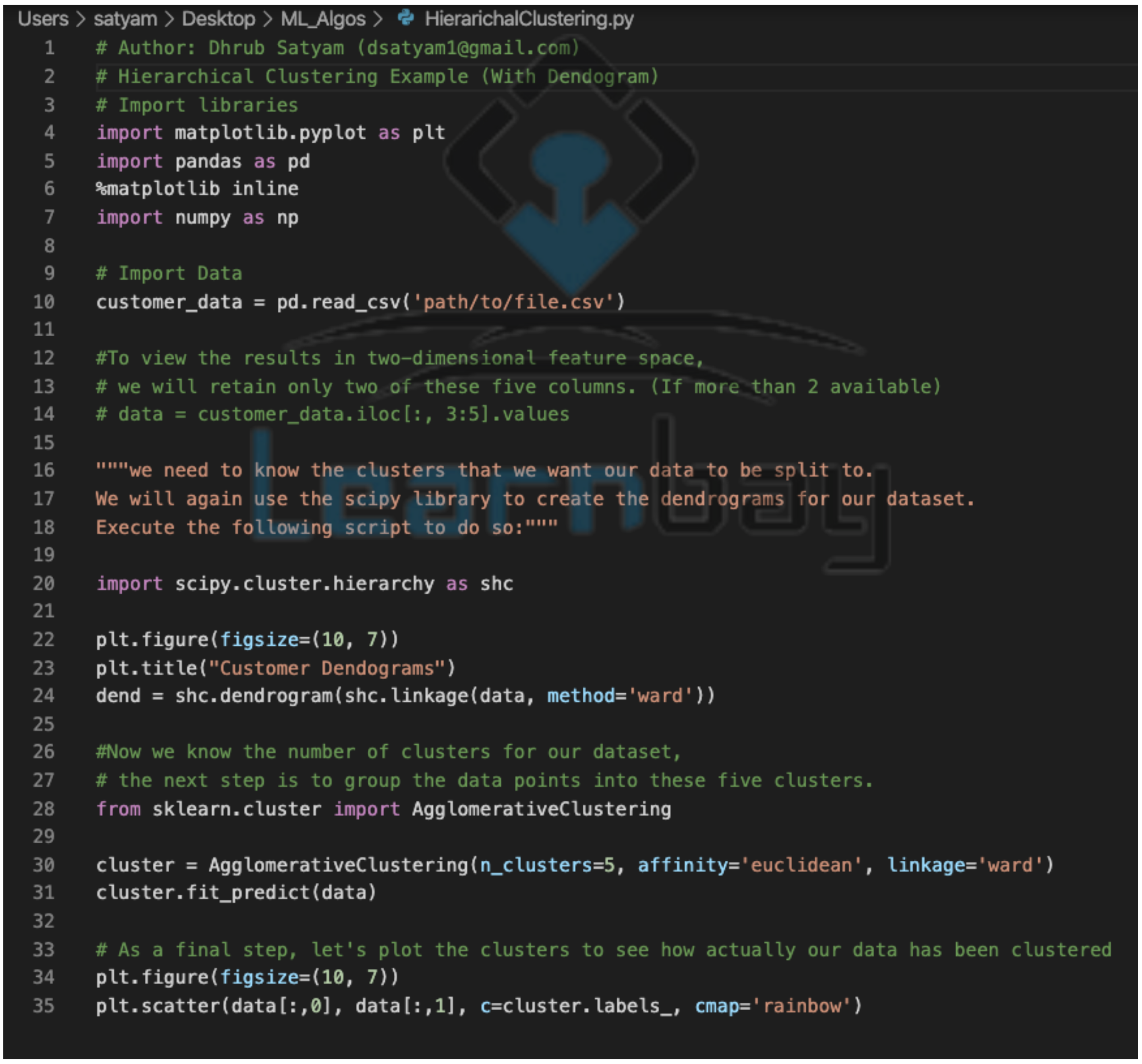
A dendrogram is a type of Tree Diagram showing hierarchical clustering — relationships between similar sets of data. They are frequently used in biology to show clustering between genes or samples, but they can represent any type of grouped data.

The role of dendrogram starts once the big cluster is formed. Dendrogram will be used to split the clusters into multiple cluster of related data points depending upon our problem.

Parts of Dendogram:



85. Hierarchical Clustering Code Snippet:


Users > satyam > Desktop > ML_Algos > HierarichalClustering.py

```python
# Author: Dhrub Satyam (dsatyam1@gmail.com)
# Hierarchical Clustering Example (With Dendogram)
# Import libraries
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline
import numpy as np

# Import Data
customer_data = pd.read_csv('path/to/file.csv')

#To view the results in two-dimensional feature space,
# we will retain only two of these five columns. (If more than 2 available)
# data = customer_data.iloc[:, 3:5].values

"""we need to know the clusters that we want our data to be split to.
We will again use the scipy library to create the dendrograms for our dataset.
Execute the following script to do so:"""

import scipy.cluster.hierarchy as shc

plt.figure(figsize=(10, 7))
plt.title("Customer Dendograms")
dend = shc.dendrogram(shc.linkage(data, method='ward'))

#Now we know the number of clusters for our dataset,
# the next step is to group the data points into these five clusters.
from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
cluster.fit_predict(data)

# As a final step, let's plot the clusters to see how actually our data has been clustered
plt.figure(figsize=(10, 7))
plt.scatter(data[:,0], data[:,1], c=cluster.labels_, cmap='rainbow')
```

.....

Gradient Boosting Algorithms

a. Adaboost
b. GBM
c. XGBoost

86. What is Boosting?

Boosting is a method of converting weak learners into strong learners. In boosting, each new tree is a fit on a modified version of the original data set.

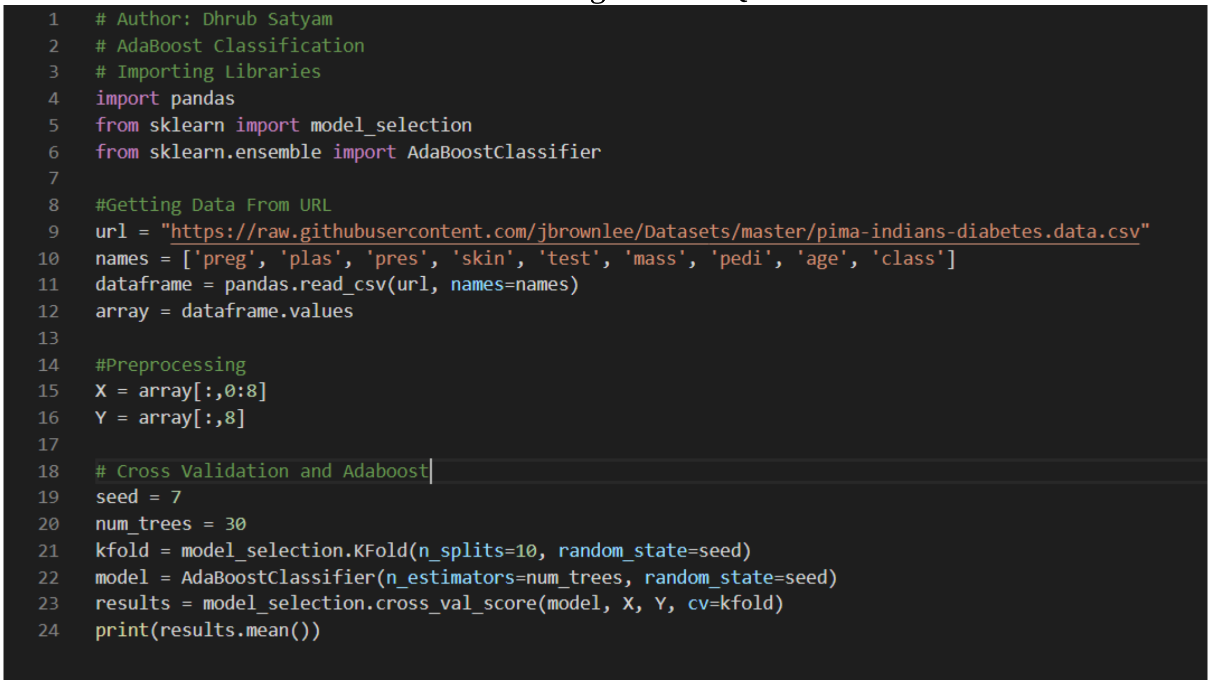
Purpose of Boosting: It helps the weak learner to be modified to become better.

How it evolved: The first Boosting Algorithm gained popularity was AdaBoost or Adaptive Boosting. Further it evolved and generalised as Gradient Boosting.

87. What is Adaboost?

Adaboost combines multiple weak learners into a single strong learner. The weak learners in AdaBoost are decision trees with a single split, called decision stumps. When AdaBoost creates its first decision stump, all observations are weighted equally. To correct the previous error, the observations that were incorrectly classified now carry more weight than the observations that were correctly classified. AdaBoost algorithms can be used for both classification and regression problem.

88. Adaboost Code Snippet:



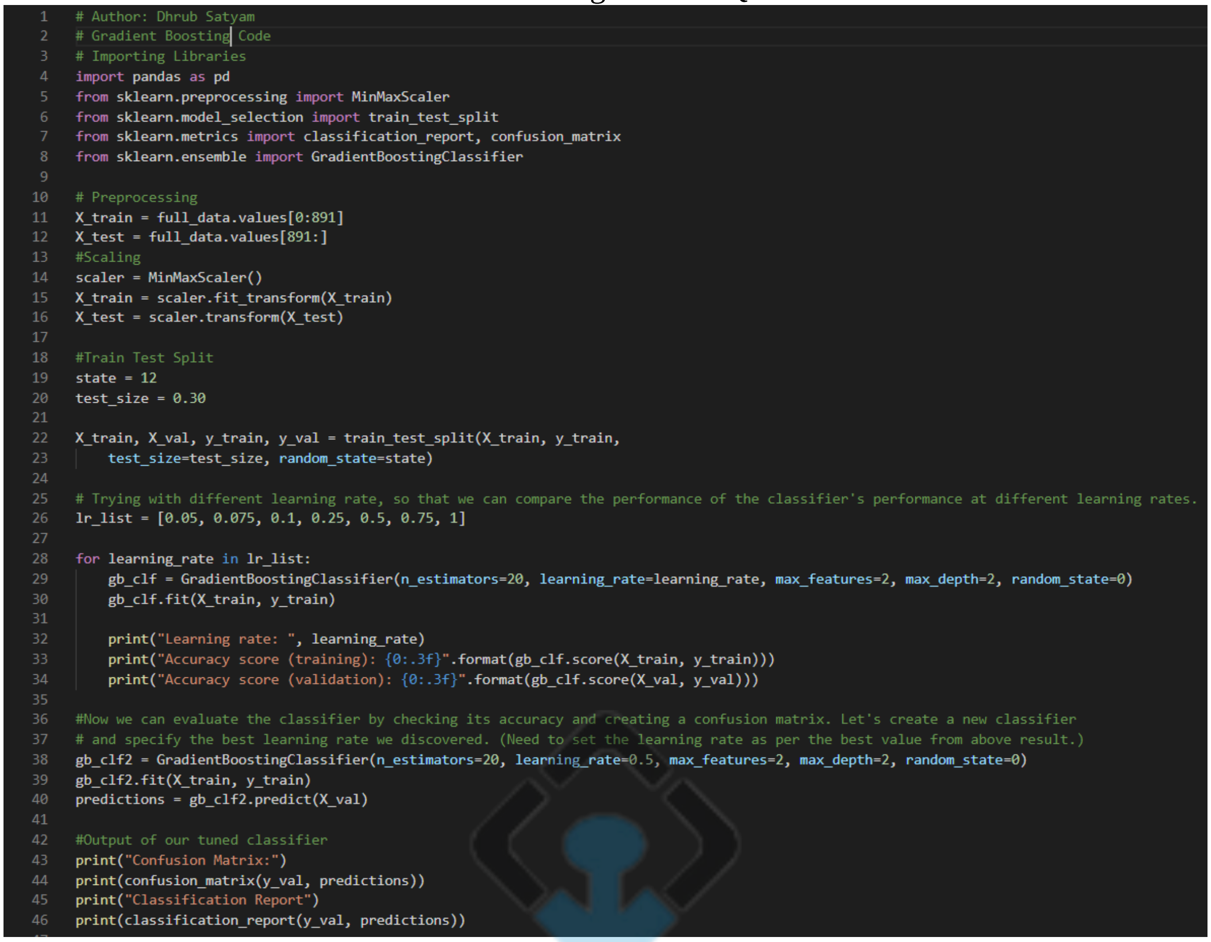
```python
# Author: Dhrub Satyam
# AdaBoost Classification
# Importing Libraries
import pandas
from sklearn import model_selection
from sklearn.ensemble import AdaBoostClassifier

#Getting Data From URL
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values

#Preprocessing
X = array[:,0:8]
Y = array[:,8]

# Cross Validation and Adaboost
seed = 7
num_trees = 30
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X, Y, cv=kfold)
print(results.mean())
```

89. What is Gradient Boosting Method (GBM)?

Gradient Boosting works by sequentially adding predictors to an ensemble, each one correcting its predecessor. However, instead of changing the weights for every incorrect classified observation at every iteration like AdaBoost, Gradient Boosting method tries to fit the new predictor to the residual errors made by the previous predictor.

GBM uses Gradient Descent to find the shortcomings in the previous learner's predictions. GBM algorithm can be given by following steps.
Fit a model to the data, F1(x) = y
Create a new model, F2(x) = F1(x) + h1(x)
By combining weak learner after weak learner, our final model is able to account for a lot of the error from the original model and reduces this error over time.

90. Gradient Boosting Code Snippet:



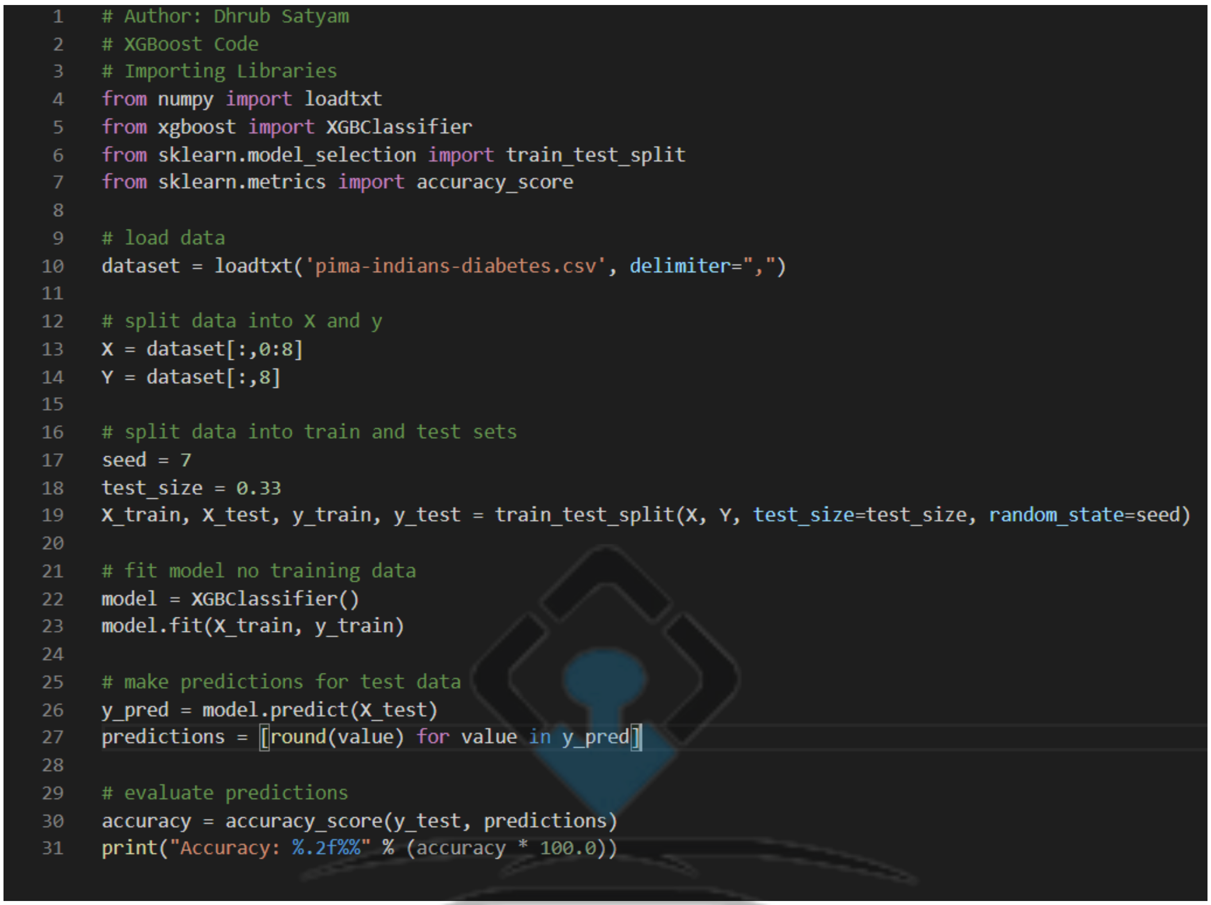
```python
# Author: Dhrub Satyam
# Gradient Boosting Code
# Importing Libraries
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.ensemble import GradientBoostingClassifier

# Preprocessing
X_train = full_data.values[0:891]
X_test = full_data.values[891:]
#Scaling
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#Train Test Split
state = 12
test_size = 0.30

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
    test_size=test_size, random_state=state)

# Trying with different learning rate, so that we can compare the performance of the classifier's performance at different learning rates.
lr_list = [0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1]

for learning_rate in lr_list:
    gb_clf = GradientBoostingClassifier(n_estimators=20, learning_rate=learning_rate, max_features=2, max_depth=2, random_state=0)
    gb_clf.fit(X_train, y_train)

    print("Learning rate: ", learning_rate)
    print("Accuracy score (training): {0:.3f}".format(gb_clf.score(X_train, y_train)))
    print("Accuracy score (validation): {0:.3f}".format(gb_clf.score(X_val, y_val)))

#Now we can evaluate the classifier by checking its accuracy and creating a confusion matrix. Let's create a new classifier
# and specify the best learning rate we discovered. (Need to set the learning rate as per the best value from above result.)
gb_clf2 = GradientBoostingClassifier(n_estimators=20, learning_rate=0.5, max_features=2, max_depth=2, random_state=0)
gb_clf2.fit(X_train, y_train)
predictions = gb_clf2.predict(X_val)

#Output of our tuned classifier
print("Confusion Matrix:")
print(confusion_matrix(y_val, predictions))
print("Classification Report")
print(classification_report(y_val, predictions))
```

91. What is XGBoost?

XGBoost stands for eXtreme Gradient Boosting. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. Gradient boosting machines are generally very slow in implementation because of sequential model training. Hence, they are not very scalable. Thus, XGBoost is focused on computational speed and model performance. XGBoost provides:

- **Parallelization** of tree construction using all of your CPU cores during training.
- **Distributed Computing** for training very large models using a cluster of machines.
- **Out-of-Core Computing** for very large datasets that don't fit into memory.
- **Cache Optimization** of data structures and algorithm to make the best use of hardware.

92. XGBoost Code Snippet:



```python
# Author: Dhrub Satyam
# XGBoost Code
# Importing Libraries
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# load data
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=",")

# split data into X and y
X = dataset[:,0:8]
Y = dataset[:,8]

# split data into train and test sets
seed = 7
test_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=test_size, random_state=seed)

# fit model no training data
model = XGBClassifier()
model.fit(X_train, y_train)

# make predictions for test data
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]

# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

93. What are basic enhancements done to Gradient Boosting?

Gradient boosting is a greedy algorithm and can overfit a training dataset quickly. It can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting.

We will look at 4 enhancements to basic gradient boosting:

1. Tree Constraints
2. Shrinkage
3. Random sampling
4. Penalized Learning

1. **Tree Constraints**: A good general heuristic is that the more constrained tree creation is, the more trees you will need in the model, and the reverse, where less constrained individual trees, the fewer trees that will be required.

Below are some constraints that can be imposed on the construction of decision trees:

Number of trees, generally adding more trees to the model can be very slow to overfit. The advice is to keep adding trees until no further improvement is observed.

Tree depth, deeper trees are more complex trees and shorter trees are preferred. Generally, better results are seen with 4-8 levels.

Number of nodes or number of leaves, like depth, this can constrain the size of the tree, but is not constrained to a symmetrical structure if other constraints are used.

Number of observations per split imposes a minimum constraint on the amount of training data at a training node before a split can be considered Minimum improvement to loss is a constraint on the improvement of any split added to a tree.

2. **Weighted Updates**: The predictions of each tree are added together sequentially. The contribution of each tree to this sum can be weighted to slow down the learning by the algorithm. This weighting is called a shrinkage or a learning rate.

3. **Stochastic Gradient Boosting**

A big insight into bagging ensembles and random forest was allowing trees to be greedily created from subsamples of the training dataset.This same benefit can be used to reduce the correlation between the trees in the sequence in gradient boosting models.This variation of boosting is called stochastic gradient boosting.

At each iteration a subsample of the training data is drawn at random (without replacement) from the full training dataset. The randomly selected subsample is then used, instead of the full sample, to fit the base learner.

4. **Penalized Gradient Boosting**

Additional constraints can be imposed on the parameterized trees in addition to their structure.

Classical decision trees like CART are not used as weak learners, instead a modified form called a regression tree is used that has numeric values in the leaf nodes (also called terminal nodes). The values in the leaves of the trees can be called weights in some literature.As such, the leaf weight values of the trees can be regularized using popular regularization functions, such as:

L1 regularization of weights.
L2 regularization of weights.

The additional regularization term helps to smooth the final learnt weights to avoid over-fitting. Intuitively, the regularized objective will tend to select a model employing simple and predictive functions.

…..
Dimensionality Reduction Algorithms/Techniques

94. What is Dimensionality Reduction? Why is it used?

Dimensionality reduction refers to the process of converting a set of data. That data needs to having vast dimensions into data with lesser dimensions. Also, it needs to ensure that it conveys similar information concisely.

Although, we use these techniques to solve machine learning problems. And problem is to obtain better features for a classification or regression task.

95. What are the commonly used Dimensionality Reduction Techniques?

The various methods used for dimensionality reduction include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)

96. How does PCA work? When to use?

Short Answer: Principal Component Analysis (PCA) is an unsupervised, non-parametric statistical technique primarily used for dimensionality reduction in machine learning.

High dimensionality means that the dataset has a large number of features. The primary problem associated with high dimensionality in the machine learning field is model overfitting, which reduces the ability to generalize beyond the examples in the training set.

PCA in Layman's Term: Consider the 2D XY plane.

For the sake of intuition, let us consider variance as the spread of data - distance between the two farthest points.

Assumption:
Typically, it is believed, that if the variance of data is large, it offers more information, than data which has small variance. (This may or may not be true). This is the assumption which PCA intends to exploit.

I give you 4 points - {(1,1), (2,2), (3,3), (4,4)}
(all lie on the line X=Y)

What is the variance on X-axis?
Variance(X) = 4-1 = 3

What is the variance on Y-axis?
Variance(Y) = 4-1 = 3

Can we obtain new data with higher variance in some manner?
Rotate your XY system by 45 degrees anticlockwise. What happens? The line X=Y has now become the X(new)-axis. And, X = -Y is now the Y(new)-axis. Let's compute the variance again (in the form of distance)

Variance(X(new)) = distance ((4,4), (1,1)) = sqrt(18) = 4.24
Variance(Y(new)) =requires some calculations.

What did we get by doing this rotation?
Original data - had highest variance on any axis as 3. This rotation gave us a variance of 4.24

That was the intuitive explanation of what PCA does. Just for further clarification

Eigenvalues = variance of the data along a particular axis in the new coordinate system. In above example, Eigenvalue(X(new)) = 4.24.

Eigenvectors = the vectors which represent the new coordinate system. In above example, vector [1,1], would be an eigenvector for X(new), and [1,-1] eigenvector for Y(new). Since they are just directions - solvers typically give us unit vectors.

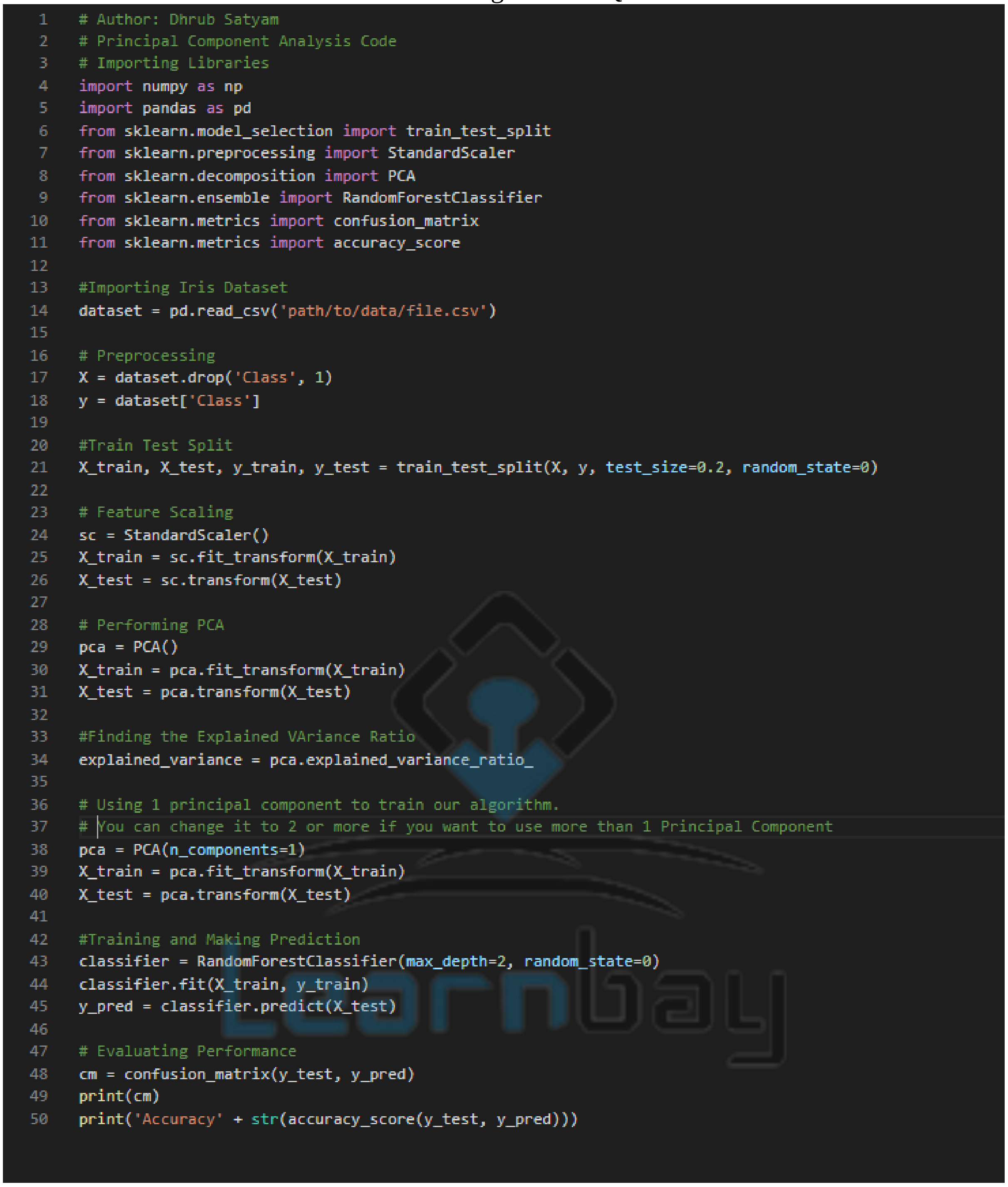
Getting transformed data
Once you have the eigenvectors, a dot product of the eigenvector with the original point will give you the new point in the new coordinate system.

Diagnolization: This is the part where you equate covariance to lambda*I. This is basically trying to find an eigenvector, such that all points would lie on the same line, and thus it will have only elements of variance, and covariance terms would be zero.

Steps of PCA:

1. Calculate the covariance matrix X of data points.
2. Calculate eigenvectors and correspond eigenvalues.
3. Sort eigenvectors accordingly to their given value in decrease order.
4. Choose first k eigenvectors and that will be the new k dimensions.
5. Transform the original n-dimensional data points into k-dimensions

97. PCA Code Snippet:



```python
# Author: Dhrub Satyam
# Principal Component Analysis Code
# Importing Libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

#Importing Iris Dataset
dataset = pd.read_csv('path/to/data/file.csv')

# Preprocessing
X = dataset.drop('Class', 1)
y = dataset['Class']

#Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Feature Scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Performing PCA
pca = PCA()
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

#Finding the Explained VAriance Ratio
explained_variance = pca.explained_variance_ratio_

# Using 1 principal component to train our algorithm.
# You can change it to 2 or more if you want to use more than 1 Principal Component
pca = PCA(n_components=1)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

#Training and Making Prediction
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Evaluating Performance
cm = confusion_matrix(y_test, y_pred)
print(cm)
print('Accuracy' + str(accuracy_score(y_test, y_pred)))
```

98. How does LDA work? When to use?

LDA is a way to reduce 'dimensionality' while at the same time preserving as much of the class discrimination information as possible.

How does it work?

Basically, LDA helps you find the 'boundaries' around clusters of classes. It projects your data points on a line so that your clusters 'are as separated as possible', with each cluster having a relative (close) distance to a centroid. What was that stuff about dimensionality?

Let's say you have a group of data points in 2 dimensions, and you want to group them into 2 groups. LDA reduces the dimensionality of your set like so: K(Groups) = 2. 2-1 = 1.

Why? Because "The K centroids lie in an at most K-1-dimensional affine subspace". What is the affine subspace? It’s a geometric concept or *structure* that says, "I am going to generalize the affine properties of Euclidean space". What are those affine properties of the Euclidean space? Basically, it’s the fact that we can represent a point with 3 coordinates in a 3-dimensional space (with a nod toward the fact that there may be more than 3 dimensions that we are ultimately dealing with).

So, we should be able to represent a point with 2 coordinates in 2 dimensional space and represent a point with 1 coordinate in a 1 dimensional space. LDA reduced our
dimensionality of our 2-dimension problem down to one dimension. So now we can get down to the serious business of listening to the data. We now have 2 groups, and 2 points in any dimension can be joined by a line. How many dimensions does a line have? 1! Now we are cooking with Crisco!
So we get a bunch of these data points, represented by their 2d representation (x,y). We are going to use LDA to group these points into either group 1 or group 2.

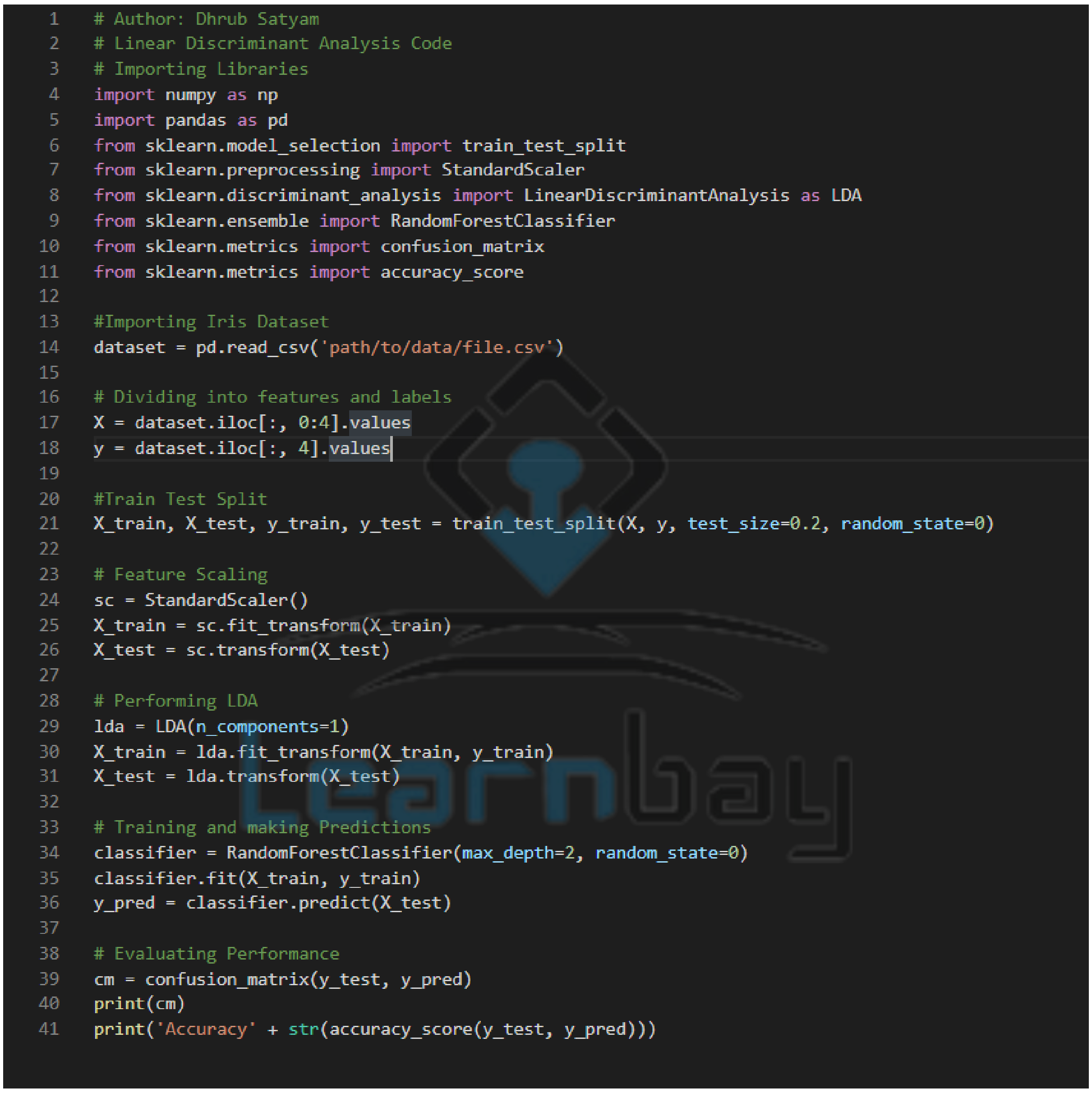
99. What are the Steps for LDA?

Steps of LDA:

1. Compute the d-dimensional mean vector for the different classes from the dataset.
2. Compute the Scatter matrix (in between class and within the class scatter matrix)
3. Sort the Eigen Vector by decrease Eigen Value and choose k eigenvector with the largest eigenvalue to from a d x k dimensional matrix w (where every column represents an eigenvector)
4. Used **d * k** eigenvector matrix to transform the sample onto the new subspace.

This can be summarized by the matrix multiplication.
Y = X x W (where X is a **n * d** dimension matrix representing the n samples and **you** are transformed **n * k** dimensional samples in the new subspace.

100. LDA Code Snippet:



```python
# Author: Dhrub Satyam
# Linear Discriminant Analysis Code
# Importing Libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

#Importing Iris Dataset
dataset = pd.read_csv('path/to/data/file.csv')

# Dividing into features and labels
X = dataset.iloc[:, 0:4].values
y = dataset.iloc[:, 4].values

#Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Feature Scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Performing LDA
lda = LDA(n_components=1)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)

# Training and making Predictions
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Evaluating Performance
cm = confusion_matrix(y_test, y_pred)
print(cm)
print('Accuracy' + str(accuracy_score(y_test, y_pred)))
```

101. What is GDA?

When we have a classification problem in which the input features are continuous random variable, we can use GDA, it's a generative learning algorithm in which we assume p(x|y) is distributed according to a **multivariate normal distribution** and p(y) is distributed according to **Bernoulli**.

Gaussian discriminant analysis (GDA) is a generative model for classification where the distribution of each class is modeled as a multivariate Gaussian.

102. What are advantages and disadvantages of Dimensionality Reduction?

Advanatges:

- Dimensionality Reduction helps in data compression, and hence reduced storage space.
- It reduces computation time.
- It also helps remove redundant features, if any.
- Dimensionality Reduction helps in data compressing and reducing the storage space required
- It fastens the time required for performing same computations.
- If there present fewer dimensions then it leads to less computing. Also, dimensions can allow usage of algorithms unfit for a large number of dimensions.
- It takes care of multicollinearity that improves the model performance. It removes redundant features. For example, there is no point in storing a value in two different units (meters and inches).
- Reducing the dimensions of data to 2D or 3D may allow us to plot and visualize it precisely. You can then observe patterns more clearly.

Disadvantages:

- Basically, it may lead to some amount of data loss.
- Although, PCA tends to find linear correlations between variables, which is sometimes undesirable.
- Also, PCA fails in cases where mean and covariance are not enough to define datasets.
- Further, we may not know how many principal components to keep- in practice, some thumb rules are applied.

.....