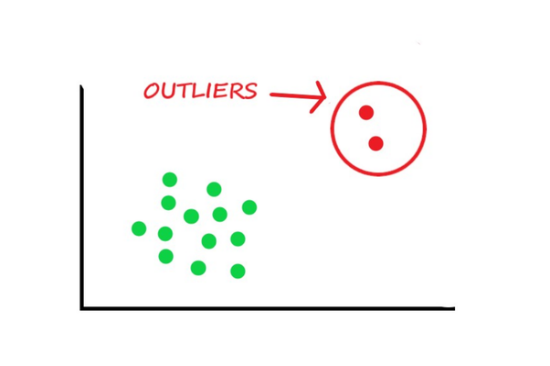
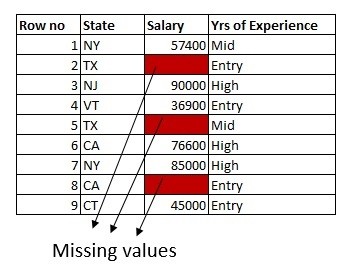
1. Differentiate a missing value with an outlier

**Outlier is the value far from the main group.** **Missing value is the value of blank**. We often meet them when we analyze large size data. Outlier and missing value are also called "abnormal value", "noise", "trash", "bad data" and "incomplete data".



2. Define Ensemble Learning.

Ensemble learning is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem.

Ensemble learning is primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unfortunate selection of a poor one. Other applications of ensemble learning include assigning a confidence to the decision made by the model, selecting optimal (or near optimal) features, data fusion, incremental learning, nonstationary learning and error-correcting.

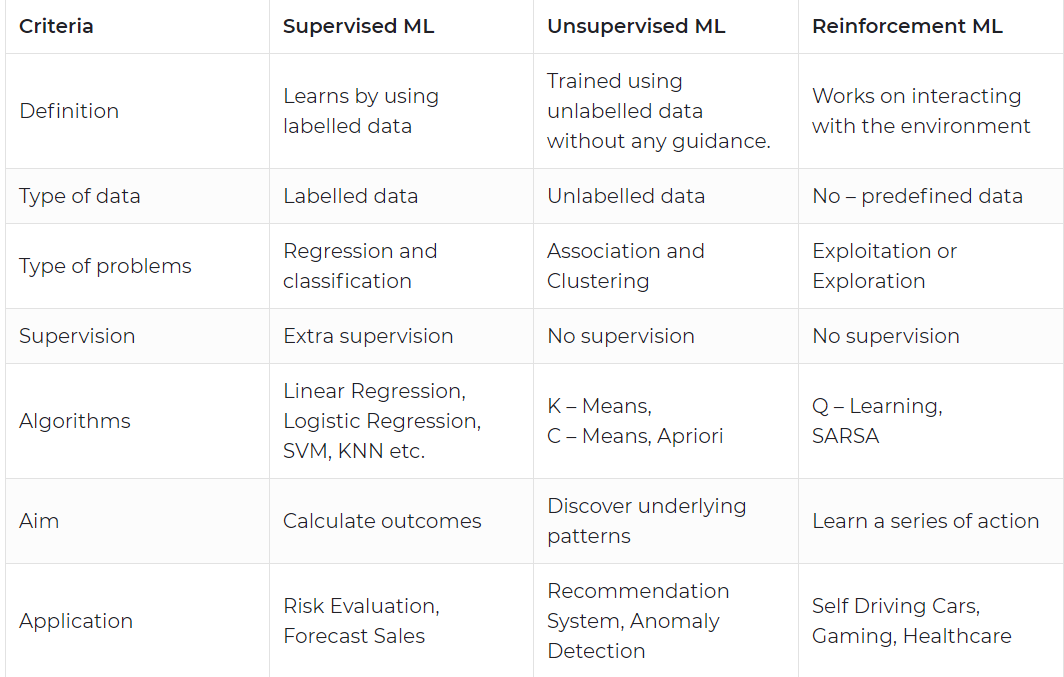
Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers (experts) and to allow them to vote.

Advantage: Improvement in predictive accuracy.  
Disadvantage: It is difficult to understand an ensemble of classifiers.

3. List out various advantages of using Decision Trees

* Compared to other algorithms decision trees requires less effort for data preparation during pre-processing.
* A decision tree does not require normalization of data.
* A decision tree does not require scaling of data as well.
* Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent.
* A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders.

4. Differentiate Unsupervised Learning and Reinforcement Learning



5. Define any problems using Naive Bayes for Classification

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes’ Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

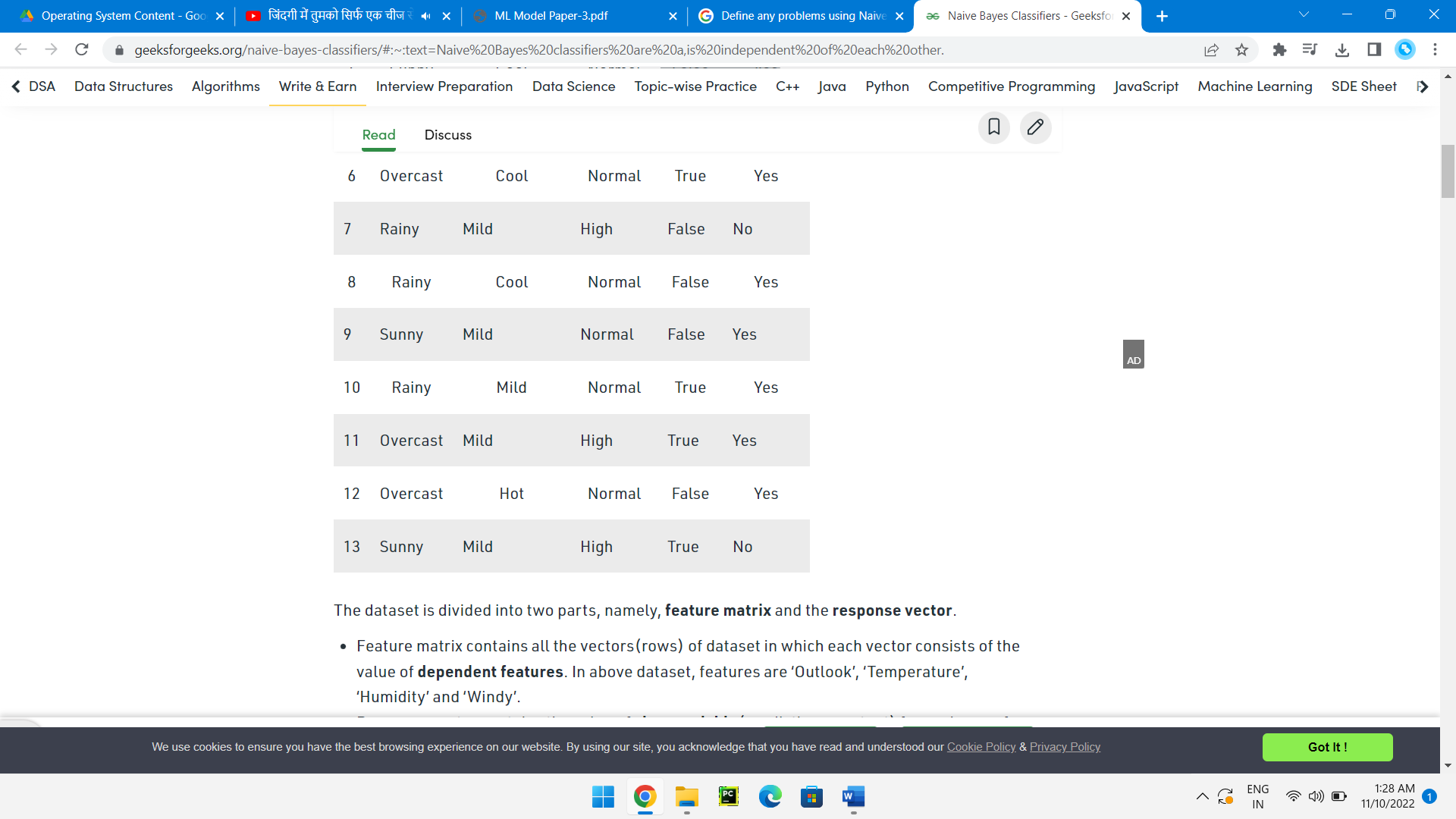
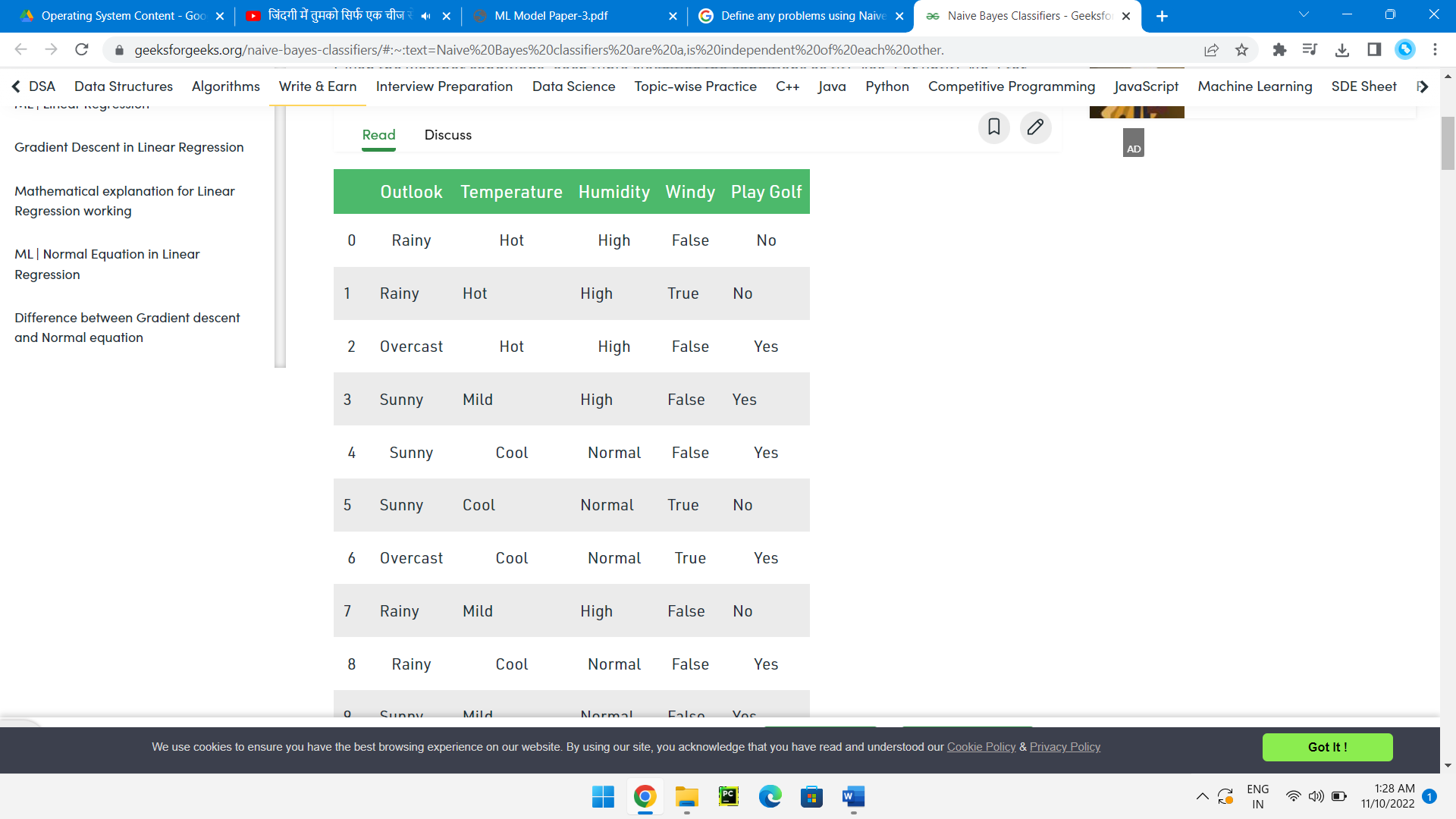
* Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are ‘Outlook’, ‘Temperature’, ‘Humidity’ and ‘Windy’.
* Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In above dataset, the class variable name is ‘Play golf’.

**Assumption:**

The fundamental Naive Bayes assumption is that each feature makes an:

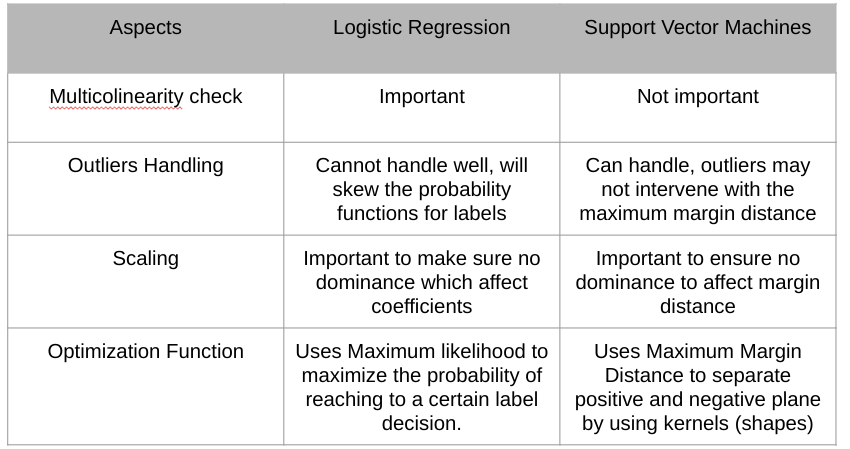
* independent
* equal

contribution to the outcome.



6. Compare SVM and logistic regression in handling outliers.

SVM supports both linear and non-linear solutions using kernel trick. SVM handles outliers better than LR. Both perform well when the training data is less, and there are large number of features.



7. Is Feature Scaling required for the KNN Algorithm? Explain with proper justification.

Yes, feature scaling is required to get the better performance of the KNN algorithm.

For Example, Imagine a dataset having n number of instances and N number of features. There is one feature having values ranging between 0 and 1. Meanwhile, there is also a feature that varies from -999 to 999. When these values are substituted in the formula of Euclidean Distance, this will affect the performance by giving higher weightage to variables having a higher magnitude.

KNN and K-Means are one of the most commonly and widely used machine learning algorithms. KNN is a supervised learning algorithm and can be used to solve both classification as well as regression problems. K-Means, on the other hand, is an unsupervised learning algorithm which is widely used to cluster data into different groups.

One thing which is common in both these algorithms is that both KNN and K-Means are distance based algorithms. [KNN](https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/) chooses the k closest neighbors and then based on these neighbors, assigns a class (for classification problems) or predicts a value (for regression problems) for a new observation. [K-Means](https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/) clusters the similar points together. The similarity here is defined by the distance between the points. Lesser the distance between the points, more is the similarity and vice versa.

8. Explain how the Random Forests give output for Classification and Regression problems

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

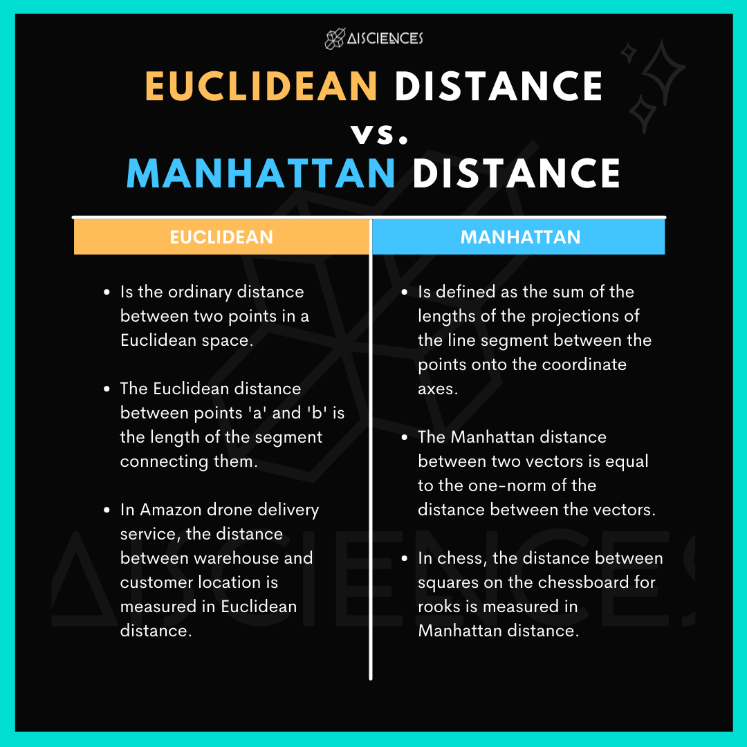
Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

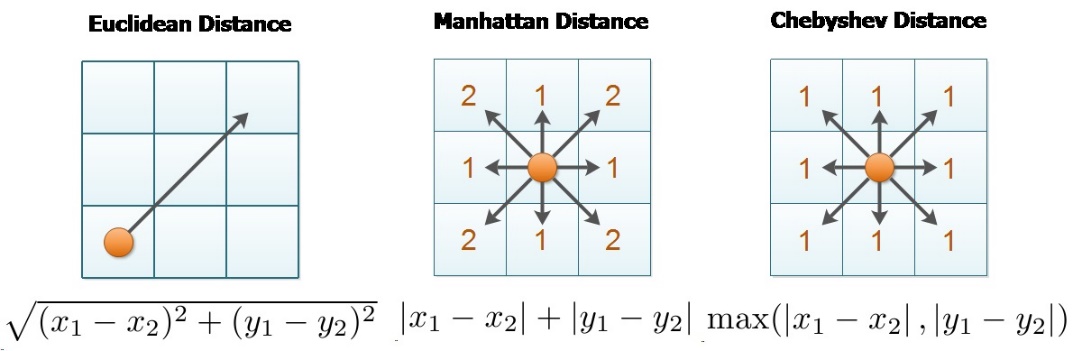
The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

A random forest regressor. A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

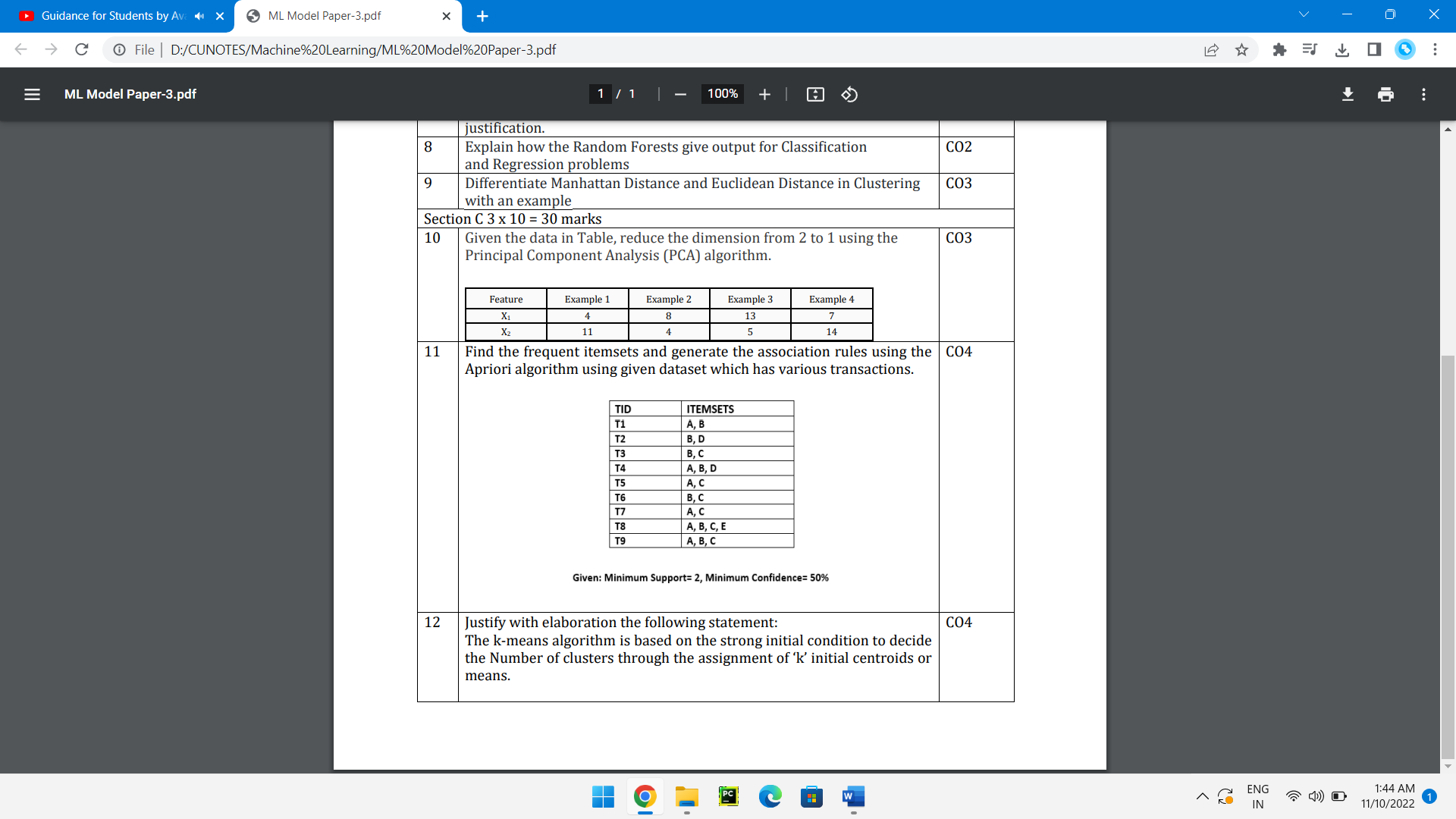
In addition to classification, Random Forests can also be used for regression tasks. A Random Forest's nonlinear nature can give it a leg up over linear algorithms, making it a great option. However, it is important to know your data and keep in mind that a Random Forest can't extrapolate.

9. Differentiate Manhattan Distance and Euclidean Distance.





Q10



### **https://www.gatevidyalay.com/principal-component-analysis-dimension-reduction/**

### ****PCA Algorithm-****

The steps involved in PCA Algorithm are as follows-

**Step-01:** Get data.

**Step-02:** Compute the mean vector (µ).

**Step-03:** Subtract mean from the given data.

**Step-04:** Calculate the covariance matrix.

**Step-05:** Calculate the eigen vectors and eigen values of the covariance matrix.

**Step-06:** Choosing components and forming a feature vector.

**Step-07:** Deriving the new data set.

## **Problem-01:**

Given data = { 2, 3, 4, 5, 6, 7 ; 1, 5, 3, 6, 7, 8 }.

Compute the principal component using PCA Algorithm.

**OR**

Consider the two dimensional patterns (2, 1), (3, 5), (4, 3), (5, 6), (6, 7), (7, 8).

Compute the principal component using PCA Algorithm.

**OR**

Compute the principal component of following data-

CLASS 1

X = 2 , 3 , 4

Y = 1 , 5 , 3

CLASS 2

X = 5 , 6 , 7

Y = 6 , 7 , 8

## **Solution-**

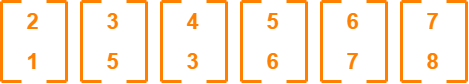
We use the above discussed PCA Algorithm-

### ****Step-01:****

Get data.

The given feature vectors are-

* x1 = (2, 1)
* x2 = (3, 5)
* x3 = (4, 3)
* x4 = (5, 6)
* x5 = (6, 7)
* x6 = (7, 8)



### ****Step-02:****

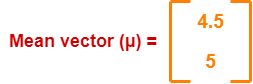
Calculate the mean vector (µ).

Mean vector (µ)

= ((2 + 3 + 4 + 5 + 6 + 7) / 6, (1 + 5 + 3 + 6 + 7 + 8) / 6)

= (4.5, 5)

Thus,



### ****Step-03:****

Subtract mean vector (µ) from the given feature vectors.

* x1 – µ = (2 – 4.5, 1 – 5) = (-2.5, -4)
* x2 – µ = (3 – 4.5, 5 – 5) = (-1.5, 0)
* x3 – µ = (4 – 4.5, 3 – 5) = (-0.5, -2)
* x4 – µ = (5 – 4.5, 6 – 5) = (0.5, 1)
* x5 – µ = (6 – 4.5, 7 – 5) = (1.5, 2)
* x6 – µ = (7 – 4.5, 8 – 5) = (2.5, 3)

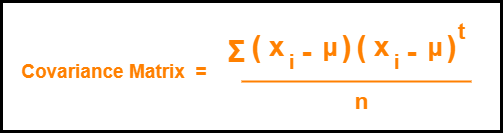
Feature vectors (xi) after subtracting mean vector (µ) are-



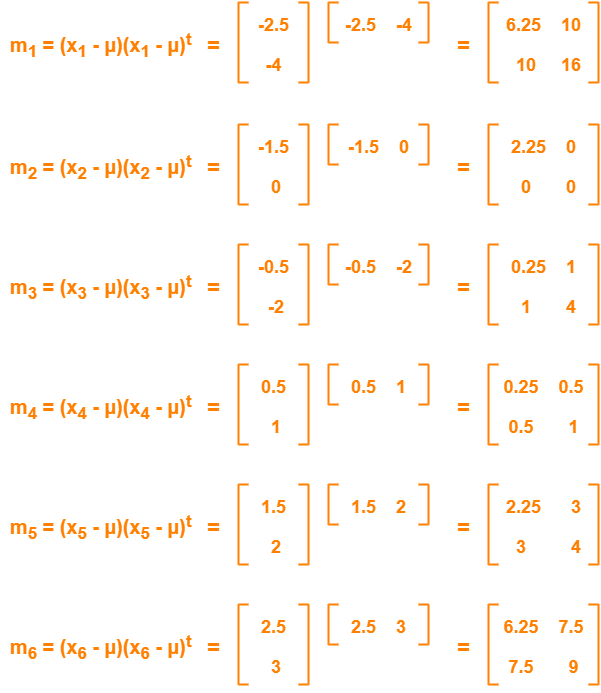
### ****Step-04:****

Calculate the covariance matrix.

Covariance matrix is given by-



Now,

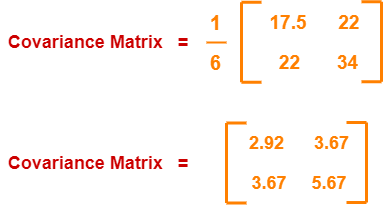


Now,

Covariance matrix

= (m1 + m2 + m3 + m4 + m5 + m6) / 6

On adding the above matrices and dividing by 6, we get-

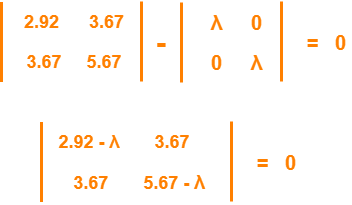


### ****Step-05:****

Calculate the eigen values and eigen vectors of the covariance matrix.

λ is an eigen value for a matrix M if it is a solution of the characteristic equation |M – λI| = 0.

So, we have-



From here,

(2.92 – λ)(5.67 – λ) – (3.67 x 3.67) = 0

16.56 – 2.92λ – 5.67λ + λ2 – 13.47 = 0

λ2 – 8.59λ + 3.09 = 0

Solving this quadratic equation, we get λ = 8.22, 0.38

Thus, two eigen values are λ1 = 8.22 and λ2 = 0.38.

Clearly, the second eigen value is very small compared to the first eigen value.

So, the second eigen vector can be left out.

Eigen vector corresponding to the greatest eigen value is the principal component for the given data set.

So. we find the eigen vector corresponding to eigen value λ1.

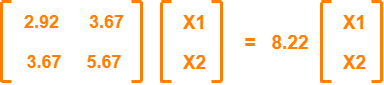
We use the following equation to find the eigen vector-

MX = λX

where-

* M = Covariance Matrix
* X = Eigen vector
* λ = Eigen value

Substituting the values in the above equation, we get-



Solving these, we get-

2.92X1 + 3.67X2 = 8.22X1

3.67X1 + 5.67X2 = 8.22X2

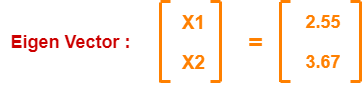
On simplification, we get-

5.3X1 = 3.67X2 ………(1)

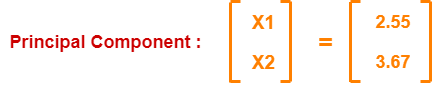
3.67X1 = 2.55X2 ………(2)

From (1) and (2), **X1 = 0.69X2**

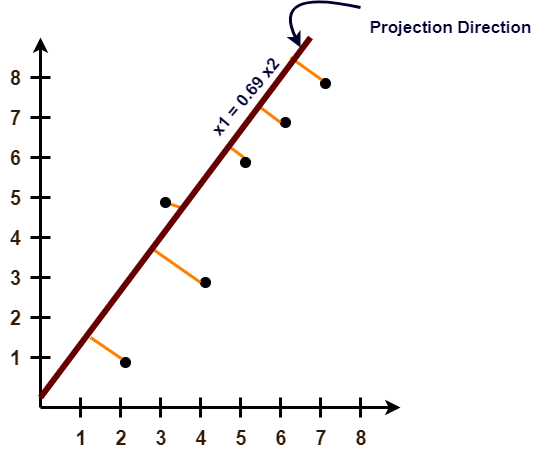
From (2), the eigen vector is-



Thus, principal component for the given data set is-

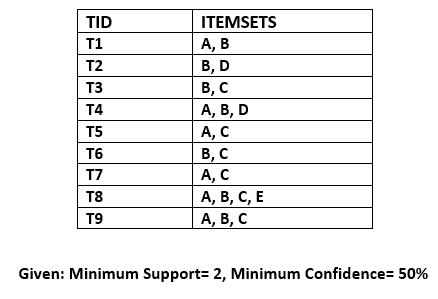


Lastly, we project the data points onto the new subspace as-



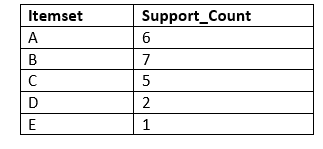
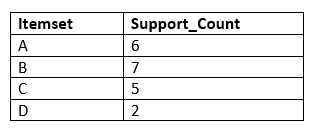
11.

**Example:** Suppose we have the following dataset that has various transactions, and from this dataset, we need to find the frequent itemsets and generate the association rules using the Apriori algorithm:

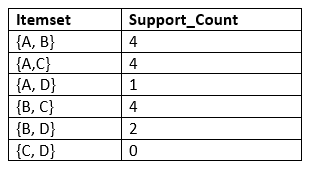
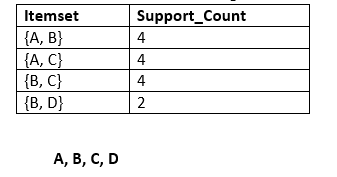


Solution:

Step-1: Calculating C1 and L1:

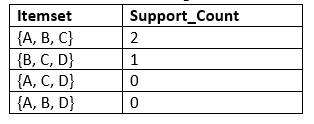
* In the first step, we will create a table that contains support count (The frequency of each itemset individually in the dataset) of each itemset in the given dataset. This table is called the **Candidate set or C1.**  
  
* Now, we will take out all the itemsets that have the greater support count that the Minimum Support (2). It will give us the table for the **frequent itemset L1.**  
  Since all the itemsets have greater or equal support count than the minimum support, except the E, so E itemset will be removed.  
  

Step-2: Candidate Generation C2, and L2:

* In this step, we will generate C2 with the help of L1. In C2, we will create the pair of the itemsets of L1 in the form of subsets.
* After creating the subsets, we will again find the support count from the main transaction table of datasets, i.e., how many times these pairs have occurred together in the given dataset. So, we will get the below table for C2:  
  
* Again, we need to compare the C2 Support count with the minimum support count, and after comparing, the itemset with less support count will be eliminated from the table C2. It will give us the below table for L2  
  

AD

Step-3: Candidate generation C3, and L3:

* For C3, we will repeat the same two processes, but now we will form the C3 table with subsets of three itemsets together, and will calculate the support count from the dataset. It will give the below table:  
  
* Now we will create the L3 table. As we can see from the above C3 table, there is only one combination of itemset that has support count equal to the minimum support count. So, the L3 will have only one combination, i.e., **{A, B, C}.**

Step-4: Finding the association rules for the subsets:

To generate the association rules, first, we will create a new table with the possible rules from the occurred combination {A, B.C}. For all the rules, we will calculate the Confidence using formula **sup( A ^B)/A.** After calculating the confidence value for all rules, we will exclude the rules that have less confidence than the minimum threshold(50%).

Consider the below table:

|  |  |  |
| --- | --- | --- |
| **Rules** | **Support** | **Confidence** |
| A ^B → C | 2 | Sup{(A ^B) ^C}/sup(A ^B)= 2/4=0.5=50% |
| B^C → A | 2 | Sup{(B^C) ^A}/sup(B ^C)= 2/4=0.5=50% |
| A^C → B | 2 | Sup{(A ^C) ^B}/sup(A ^C)= 2/4=0.5=50% |
| C→ A ^B | 2 | Sup{(C^( A ^B)}/sup(C)= 2/5=0.4=40% |
| A→ B^C | 2 | Sup{(A^( B ^C)}/sup(A)= 2/6=0.33=33.33% |
| B→ B^C | 2 | Sup{(B^( B ^C)}/sup(B)= 2/7=0.28=28% |

As the given threshold or minimum confidence is 50%, so the first three rules **A ^B → C, B^C → A, and A^C → B** can be considered as the strong association rules for the given problem.

12.

K-Means Clustering is an [Unsupervised Learning algorithm](https://www.javatpoint.com/unsupervised-machine-learning), which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means [clustering](https://www.javatpoint.com/clustering-in-machine-learning) algorithm mainly performs two tasks:

* Determines the best value for K center points or centroids by an iterative process.
* Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.



Our aim here is to minimize the distance between the points within a cluster.

There is an algorithm that tries to minimize the distance of the points in a cluster with their centroid – the k-means clustering technique.

K-means is a centroid-based algorithm, or a distance-based algorithm, where we calculate the distances to assign a point to a cluster. In K-Means, each cluster is associated with a centroid.

***The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid.***

Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

Let’s now take an example to understand how K-Means actually works:

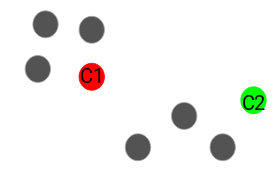
[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-09-12-21-43.png)  
We have these 8 points and we want to apply k-means to create clusters for these points. Here’s how we can do it.

Step 1: Choose the number of clusters *k*

The first step in k-means is to pick the number of clusters, k.

Step 2: Select k random points from the data as centroids

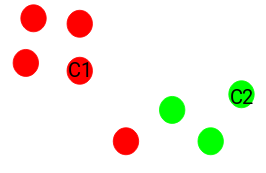
Next, we randomly select the centroid for each cluster. Let’s say we want to have 2 clusters, so k is equal to 2 here. We then randomly select the centroid:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-09-12-23-55.png)

Here, the red and green circles represent the centroid for these clusters.

Step 3: Assign all the points to the closest cluster centroid

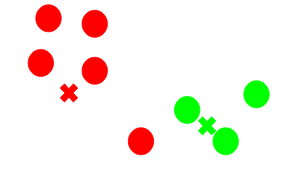
Once we have initialized the centroids, we assign each point to the closest cluster centroid:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-09-12-24-35.png)

Here you can see that the points which are closer to the red point are assigned to the red cluster whereas the points which are closer to the green point are assigned to the green cluster.

Step 4: Recompute the centroids of newly formed clusters

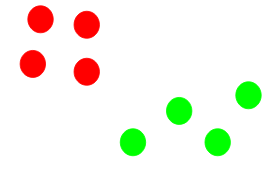
Now, once we have assigned all of the points to either cluster, the next step is to compute the centroids of newly formed clusters:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-09-12-26-59.png)

Here, the red and green crosses are the new centroids.

Step 5: Repeat steps 3 and 4

We then repeat steps 3 and 4:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-09-12-28-14.png)

*The step of computing the centroid and assigning all the points to the cluster based on their distance from the centroid is a single iteration*. But wait – when should we stop this process? It can’t run till eternity, right?

Stopping Criteria for K-Means Clustering

There are essentially three stopping criteria that can be adopted to stop the K-means algorithm:

1. Centroids of newly formed clusters do not change
2. Points remain in the same cluster
3. Maximum number of iterations are reached

We can stop the algorithm if the centroids of newly formed clusters are not changing. Even after multiple iterations, if we are getting the same centroids for all the clusters, we can say that the algorithm is not learning any new pattern and it is a sign to stop the training.

Another clear sign that we should stop the training process if the points remain in the same cluster even after training the algorithm for multiple iterations.

Finally, we can stop the training if the maximum number of iterations is reached. Suppose if we have set the number of iterations as 100. The process will repeat for 100 iterations before stopping.