Loan Prediction Based on customer behaviour

Name: Saurav Kumar

Abstract:
In this project we aim to predict possible defaulters for the consumer loans product based on the customer historic behaviour.For ex: Lets say an organisatiom wants to predict who possible defaulters are for the consumer loans product. They have data about historic customer behaviour based on what they have observed. Hence when they acquire new Customers they want to predict who is riskier and who is not.

Understanding the data

The main issue of the dataset are:

1. Preprocessing required to fill unknown values in the dataset.
2. Preprocessing required to decide on usage of categorical data along with continuous data
3. The data is class imbalanced (Number of class 1 (yes) is very low when compared to the number of Class 0 (no))

Data are from hackathon and data is not for real world. We have a prototype of data

Analysis of data:

- About 90% of all applicants are single, and only 10% are married
- About 92% of all applicants are rented, only 8% own a house;
- In all 252000 applicants, 176000 of them don't have a car, and 76000 have one (why the number is so exact?);
- There are 5806 statisticians and 5390 psychologists, ranked no.2 and no. 4 respectively.
- And in all 252 thousand applicants, there are no teachers;
- For the applicants age, there are almost same amount of applicants among 21~30 and 70~79. Do you still want to apply loans at age 75?
- In all applicants, the shortest time one lives in the current place is 10 years, the longest is 14 yrs.
- Remember that 92% applicants are rented? how come everyone rent a place at least 10 years? If you rent less than 10 years, you are not allowed to apply for a loan?

Data set

No .	Feature Name	Feature type	Feature Description
1.	Income	Numeric	Income of the client
2.	Age	Numeric	Age of the client
3.	Experience	Numeric	No of year a client has worked on a particular field
4.	Marital status	Categorical	Divided into married or unmarried

5.	House Ownership	Categorical	Divided into rented own
6.	Car ownership	Categorical	
7.	Profession	Categorical	
8	City	categorical	
9	State	categorical	

Understanding the Feature given

```
[2]:
df = pd.read_csv("../input/loan-prediction-based-on-customer-behavior/Training Data.csv", in
dex_col='Id')
df.shape
```

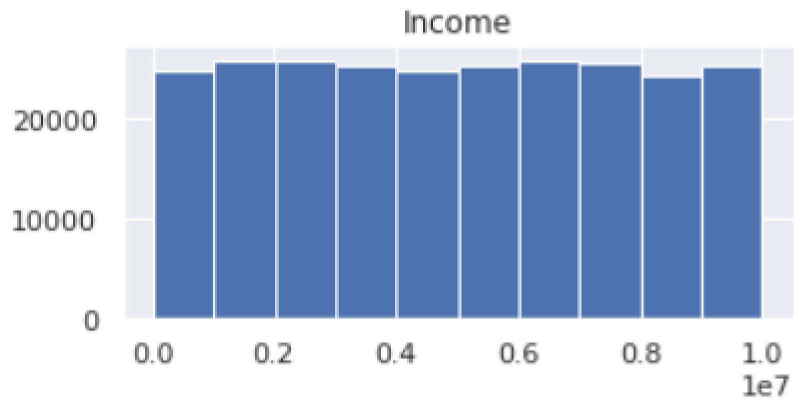
```
t[2]:
(252000, 12)
```

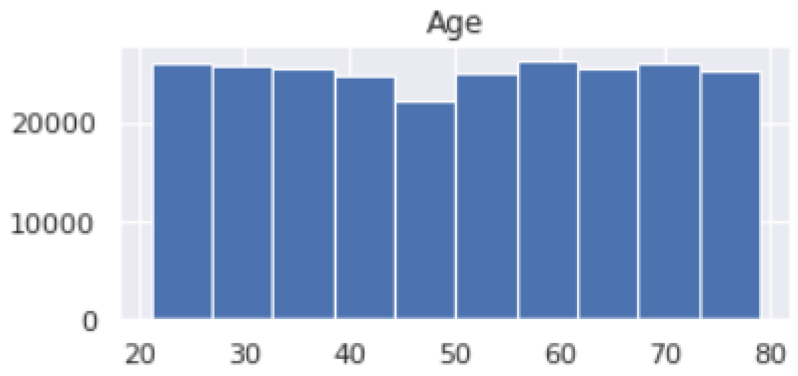
```
feats = df.columns[:-1]
is_cat = np.array([df[f].dtype == 'object' for f in feats])
cat_feats, num_feats = feats[is_cat].tolist(), feats[~is_cat].tolist()
print(cat_feats, num_feats, sep='\n')
```

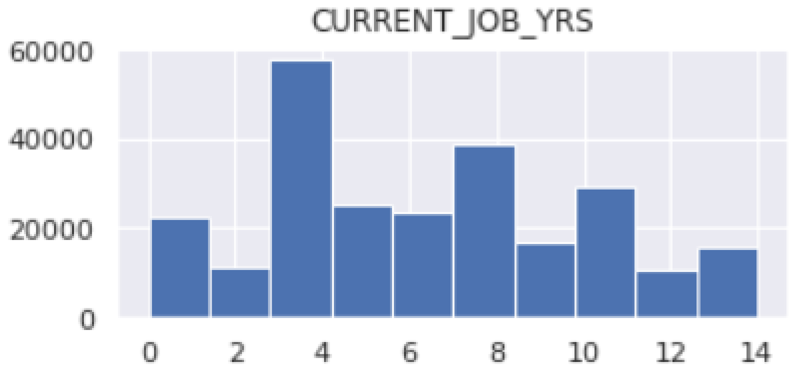
```
['Married/Single', 'House_Ownership', 'Car_Ownership', 'Profession', 'CITY', 'STATE']
['Income', 'Age', 'Experience', 'CURRENT_JOB_YRS', 'CURRENT_HOUSE_YRS']
```

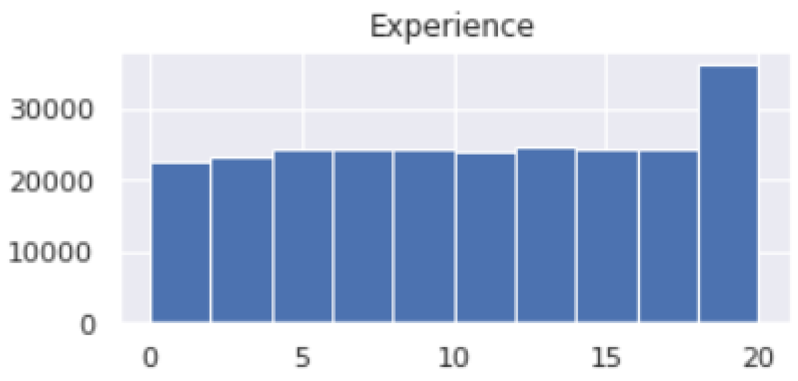
```
def show_hists(df: pd.DataFrame) -> None:
    fig = plt.figure(figsize=(12, 8))
    plt.subplots_adjust(hspace=0.5, wspace=0.5)
    for i, f in enumerate(df.columns):
        axis = fig.add_subplot(3, 2, i + 1)
        axis.hist(df[f])
        if df[f].dtype == 'object' and df[f].nunique() > 3:
            axis.set(xlabel=None, title=f)
        else:
            axis.set(title=f)
    fig.show()
```

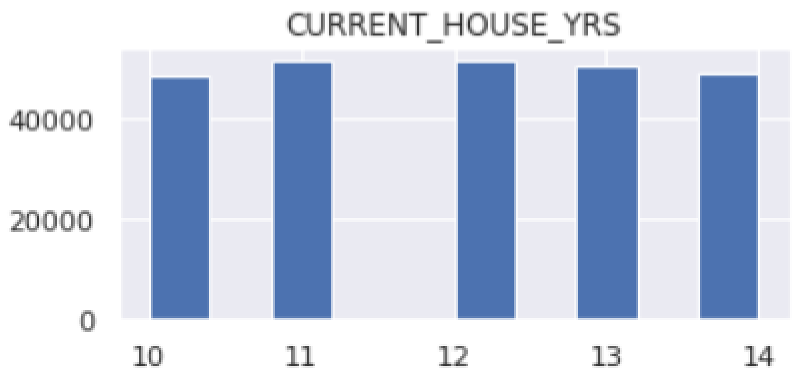
```
show_hists(df[num_feats])
show_hists(df[cat_feats[:3]])
```

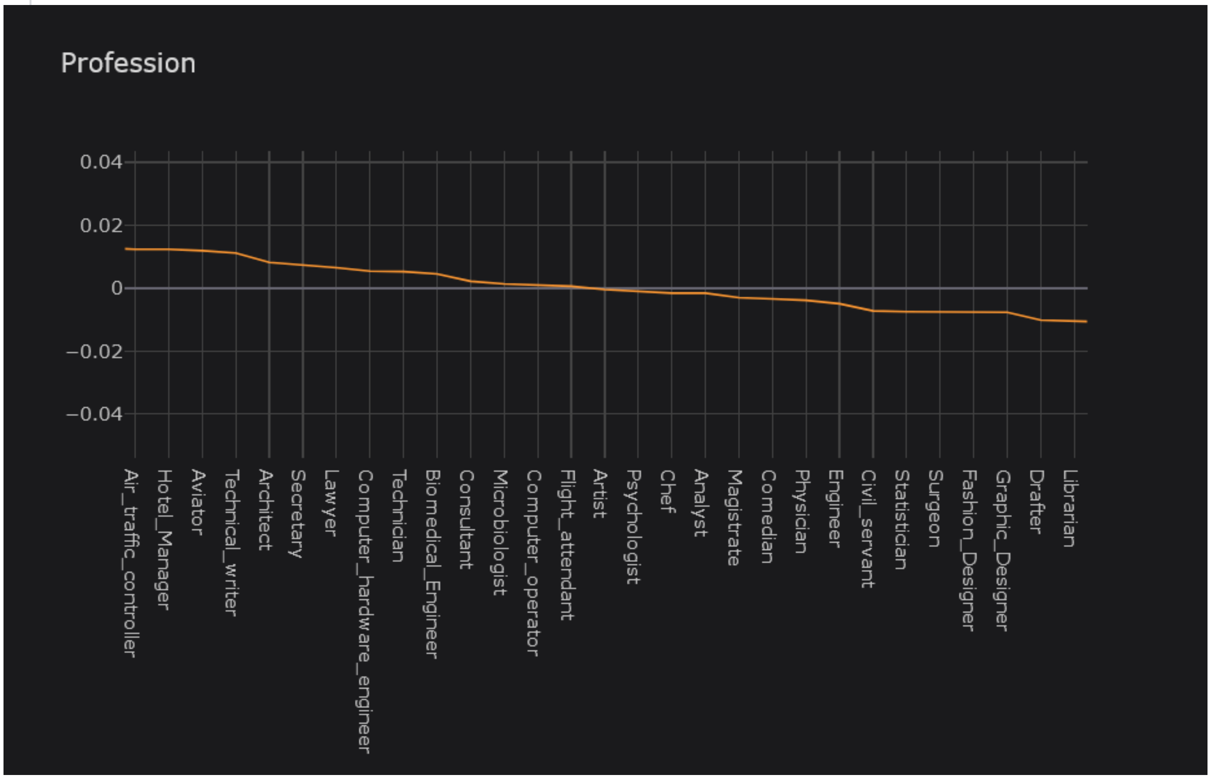

Income
0
10000
20000
0.0
0.2
0.4
0.6
0.8
1.0
1e7

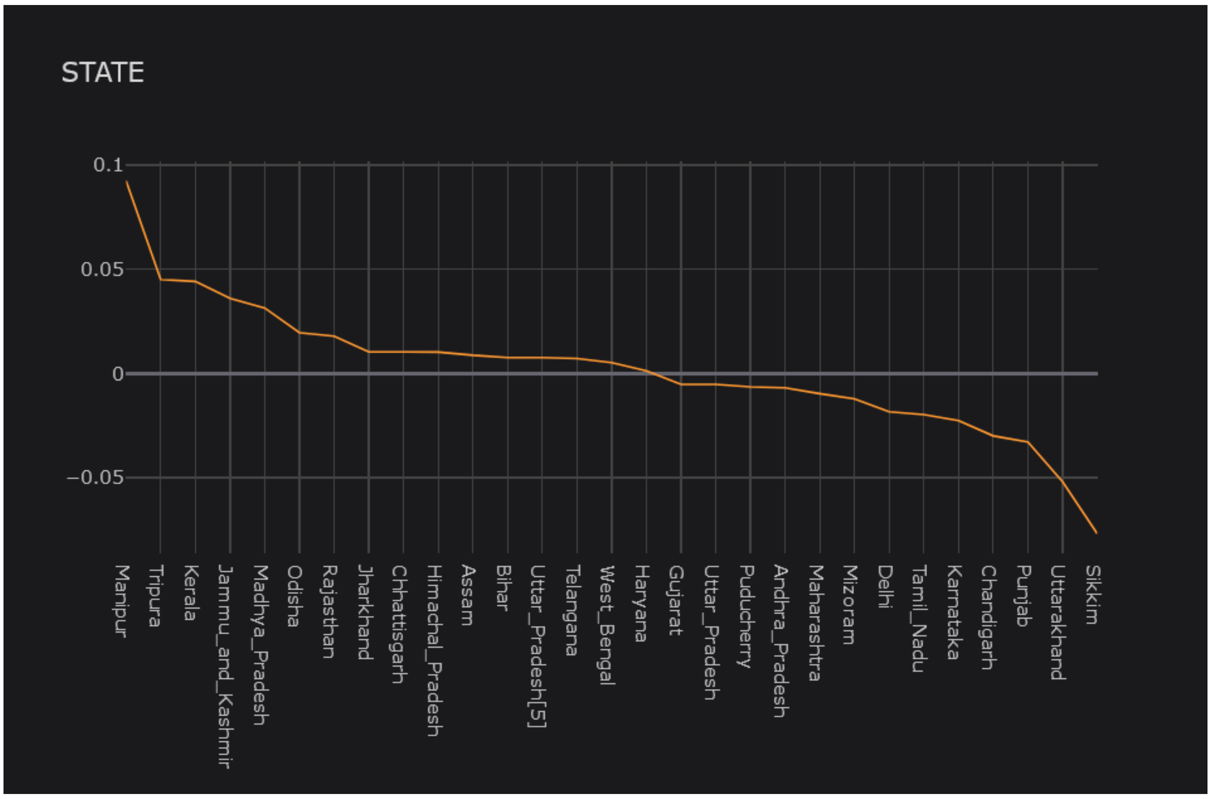

Age
0
10000
20000
20
30
40
50
60
70
80


CURRENT_JOB_YRS
0
20000
40000
60000
0
2
4
6
8
10
12
14


Experience
0
10000
20000
30000
0
5
10
15
20


CURRENT_HOUSE_YRS
40000
20000
0
10
11
12
13
14

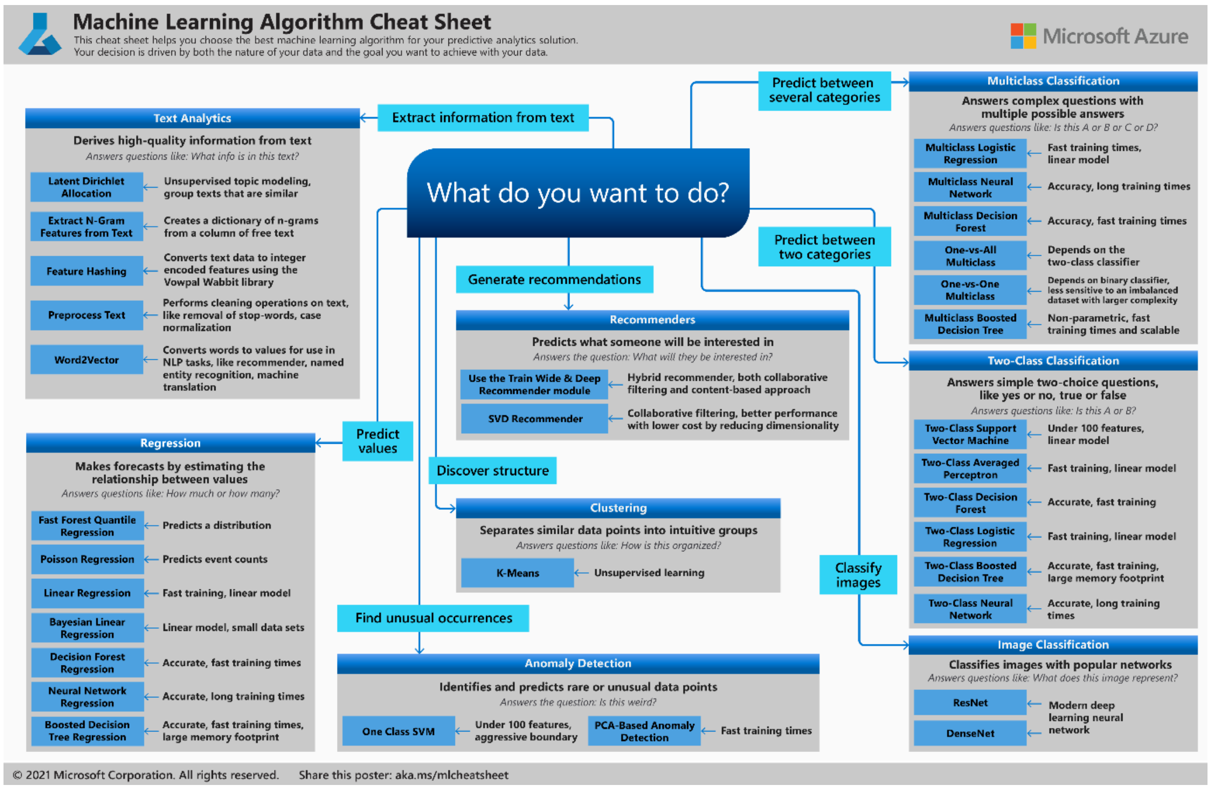

Profession
0.04
0.02
0
−0.02
−0.04
Air_traffic_controller
Hotel_Manager
Aviator
Technical_writer
Architect
Secretary
Lawyer
Computer_hardware_engineer
Technician
Biomedical_Engineer
Consultant
Microbiologist
Computer_operator
Flight_attendant
Artist
Psychologist
Chef
Analyst
Magistrate
Comedian
Physician
Engineer
Civil_servant
Statistician
Surgeon
Fashion_Designer
Graphic_Designer
Drafter
Librarian


STATE
0.1
0.05
0
−0.05
Manipur
Tripura
Kerala
Jammu_and_Kashmir
Madhya_Pradesh
Odisha
Rajasthan
Jharkhand
Chhattisgarh
Himachal_Pradesh
Assam
Bihar
Uttar_Pradesh[5]
Telangana
West_Bengal
Haryana
Gujarat
Uttar_Pradesh
Puducherry
Andhra_Pradesh
Maharashtra
Mizoram
Delhi
Tamil_Nadu
Karnataka
Chandigarh
Punjab
Uttarakhand
Sikkim

Basic Knowledge of Machine learning

Links:

Cheat sheet - https://github.com/afshinea/stanford-cs-229-machine-learning
https://www.javatpoint.com/machine-learning-random-forest-algorithm
https://www.geeksforgeeks.org/ml-types-learning-supervised-learning/
https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-cheat-sheet
https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-unsupervised-learning



Machine Learning Algorithm Cheat Sheet
This cheat sheet helps you choose the best machine learning algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the goal you want to achieve with your data.
Microsoft Azure
What do you want to do?
Extract information from text
Text Analytics
Derives high-quality information from text
Answers questions like: What info is in this text?
Latent Dirichlet Allocation — Unsupervised topic modeling, group texts that are similar
Extract N-Gram Features from Text — Creates a dictionary of n-grams from a column of free text
Feature Hashing — Converts text data to integer encoded features using the Vowpal Wabbit library
Preprocess Text — Performs cleaning operations on text, like removal of stop-words, case normalization
Word2Vector — Converts words to values for use in NLP tasks, like recommender, named entity recognition, machine translation
Predict values
Regression
Makes forecasts by estimating the relationship between values
Answers questions like: How much or how many?
Fast Forest Quantile Regression — Predicts a distribution
Poisson Regression — Predicts event counts
Linear Regression — Fast training, linear model
Bayesian Linear Regression — Linear model, small data sets
Decision Forest Regression — Accurate, fast training times
Neural Network Regression — Accurate, long training times
Boosted Decision Tree Regression — Accurate, fast training times, large memory footprint
Generate recommendations
Recommenders
Predicts what someone will be interested in
Answers the question: What will they be interested in?
Use the Train Wide & Deep Recommender module — Hybrid recommender, both collaborative filtering and content-based approach
SVD Recommender — Collaborative filtering, better performance with lower cost by reducing dimensionality
Discover structure
Clustering
Separates similar data points into intuitive groups
Answers questions like: How is this organized?
K-Means — Unsupervised learning
Find unusual occurrences
Anomaly Detection
Identifies and predicts rare or unusual data points
Answers the question: Is this weird?
One Class SVM — Under 100 features, aggressive boundary
PCA-Based Anomaly Detection — Fast training times
Predict between several categories
Multiclass Classification
Answers complex questions with multiple possible answers
Answers questions like: Is this A or B or C or D?
Multiclass Logistic Regression — Fast training times, linear model
Multiclass Neural Network — Accuracy, long training times
Multiclass Decision Forest — Accuracy, fast training times
One-vs-All Multiclass — Depends on the two-class classifier
One-vs-One Multiclass — Depends on binary classifier, less sensitive to an imbalanced dataset with larger complexity
Multiclass Boosted Decision Tree — Non-parametric, fast training times and scalable
Predict between two categories
Two-Class Classification
Answers simple two-choice questions, like yes or no, true or false
Answers questions like: Is this A or B?
Two-Class Support Vector Machine — Under 100 features, linear model
Two-Class Averaged Perceptron — Fast training, linear model
Two-Class Decision Forest — Accurate, fast training
Two-Class Logistic Regression — Fast training, linear model
Two-Class Boosted Decision Tree — Accurate, fast training, large memory footprint
Two-Class Neural Network — Accurate, long training times
Classify images
Image Classification
Classifies images with popular networks
Answers questions like: What does this image represent?
ResNet
DenseNet
Modern deep learning neural network
© 2021 Microsoft Corporation. All rights reserved. Share this poster: aka.ms/mlcheatsheet

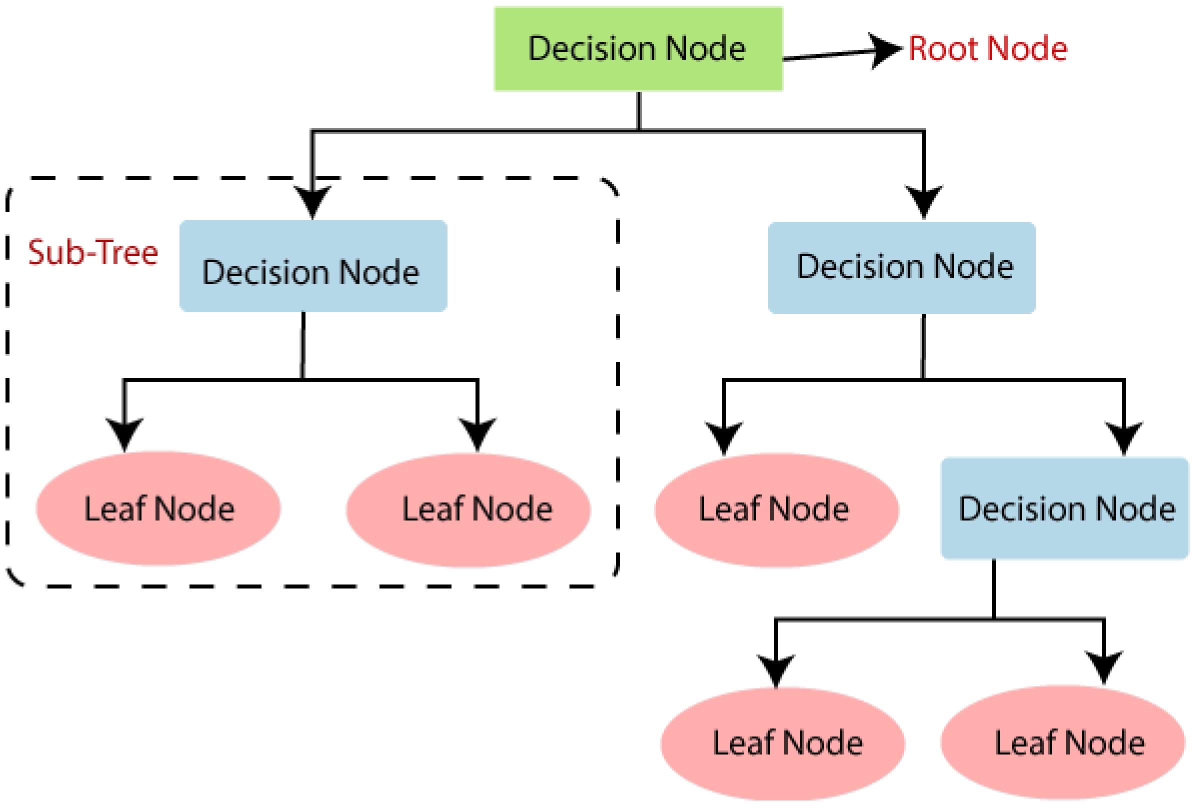
Basically three type of machine learning are

- Linear Regression
- Nearest Neighbor
- Gaussian Naive Bayes
- Decision Trees
- Support Vector Machine (SVM)
- Random Forest

Decision Tree Classification Algorithm(link)

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**

- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- **It calculates how much information a feature provides us about a class.**
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

Gini Index: It is calculated by subtracting the sum of squared probabilities of each class from one. It favors larger partitions and easy to implement whereas information gain favors smaller partitions with distinct values.

Gini Index= $1- \sum_j P_j^2$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

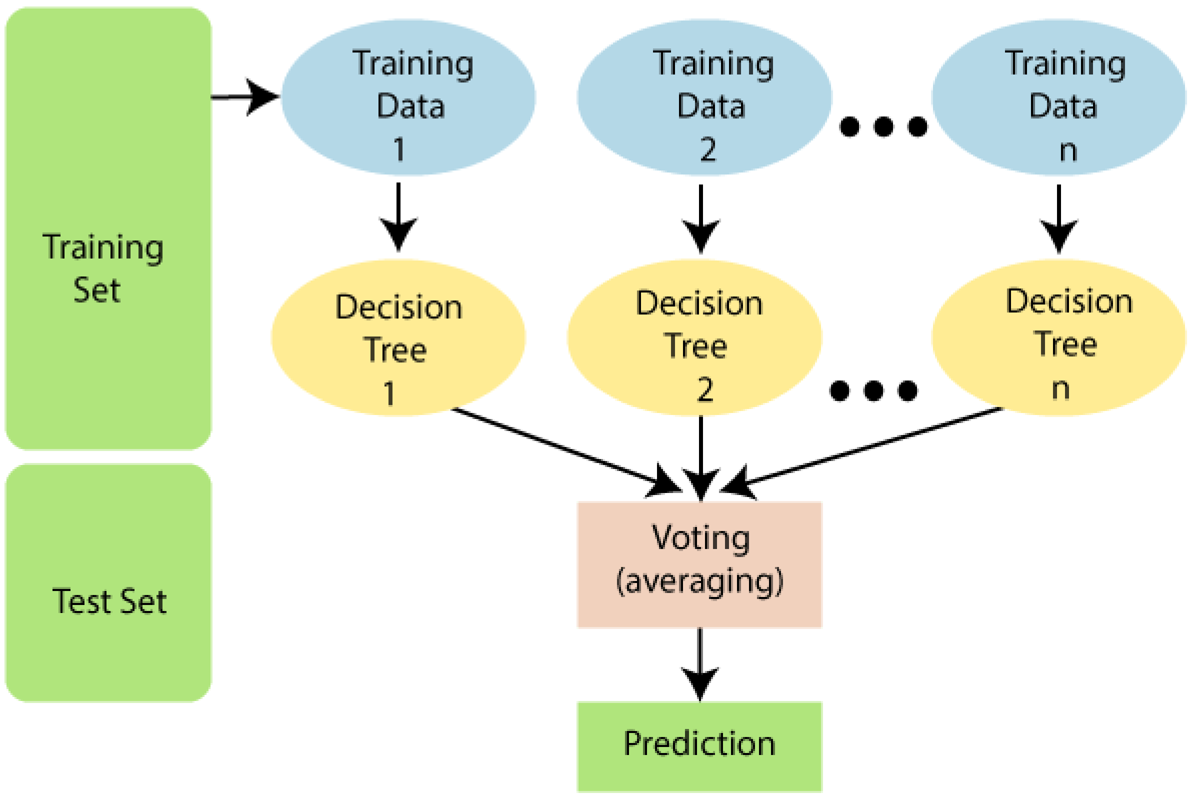
Random Forest Algorithm(link)

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, *"**Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.**"* 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 below diagram explains the working of the Random Forest algorithm:



Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

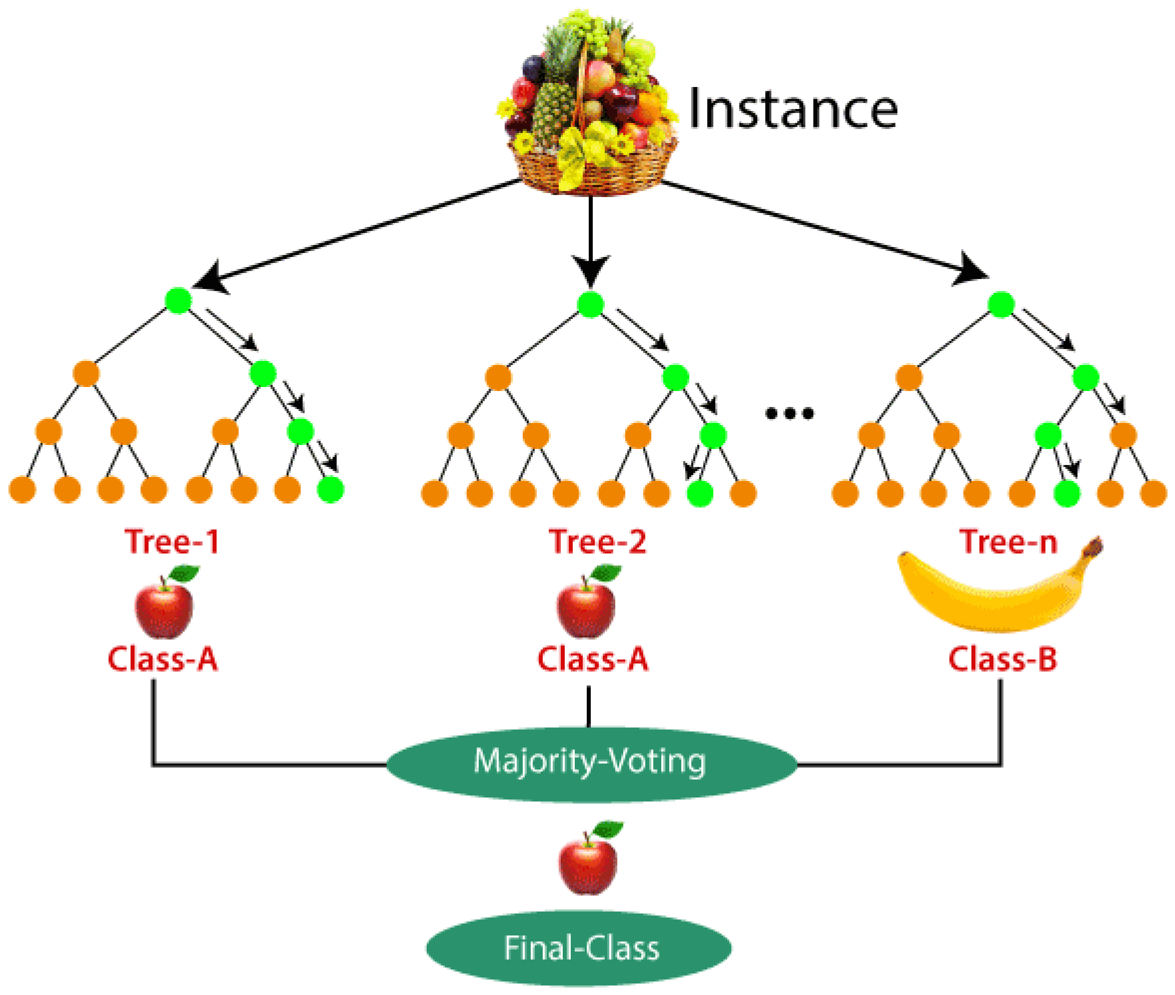
Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

Gradient descent

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

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.

Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. To find a local minimum of a function using gradient descent, we take steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. But if we instead take steps proportional to the positive of the gradient, we approach a local maximum of that function; the procedure is then known as **gradient ascent**. Gradient descent was originally proposed by Cauchy in 1847.

Gradient descent is also known as **steepest descent**; but gradient descent should not be confused with the method of steepest descent for approximating integrals.

Gradient Descent Procedure

The procedure starts off with initial values for the coefficient or coefficients for the function. These could be 0.0 or a small random value.

coefficient = 0.0

The cost of the coefficients is evaluated by plugging them into the function and calculating the cost.

cost = f(coefficient)

or

cost = evaluate(f(coefficient))

The derivative of the cost is calculated. The derivative is a concept from calculus and refers to the slope of the function at a given point. We need to know the slope so that we know the direction (sign) to move the coefficient values in order to get a lower cost on the next iteration.

delta = derivative(cost)

Now that we know from the derivative which direction is downhill, we can now update the coefficient values. A learning rate parameter (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.

XGB classifier(link)

2. Introduction to XGBoost Algorithm

Basically, XGBoost is an algorithm. Also, it has recently been dominating applied machine learning. XGBoost is an implementation of **gradient boosted** decision trees. Although, it was designed for speed and performance. Basically, it is a type of software library. That you can download and install on your machine. Then have to access it from a variety of interfaces.

XGBoost (eXtreme Gradient Boosting) is an advanced implementation of gradient boosting algorithm.

The XGBoost library implements the gradient boosting decision tree algorithm.

This algorithm goes by lots of different names such as gradient boosting, multiple additive regression trees, stochastic gradient boosting or gradient boosting machines.

Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. A popular example is the AdaBoost algorithm that weights data points that are hard to predict.

Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

This approach supports both regression and classification predictive modeling problems.

New

eXtreme Gradient Boosting (XGBoost) is a scalable and improved version of the gradient boosting algorithm *(terminology alert)* designed for efficacy, computational speed and model performance. It is an open-source library and a part of the Distributed Machine Learning Community. XGBoost is a perfect blend of software and hardware capabilities designed to enhance existing boosting techniques with accuracy in the shortest amount of time.

How did you calculate the accuracy?
what are the things you have done for precomputation