Objective:

Using the given data-set to generate a decision tree model that can successfully predict for a new applicant with recorded data for given parameters in the data set, if he is likely to be a defaulter.

Project on AI Powered Credit Risk Analysis

Topic Covered:

- **What is Credit Risk Analysis and Why it is Important?**
- **ML/AI in Credit Risk Analysis**
- **Decision Tree, its Advantages and Disadvantages**
- **Description of various Python modules used in bulding Decision Tree Clasifier and in visualizing a decision tree**
- **Plot of the decision tree obtained**
- **Accuracy of the model via gini index and entropy criterion**

What is Credit Risk Analysis?

Credit risk analysis is assessing the possibility of the borrower's repayment failure and the loss caused to the financer when the borrower does not for any reason repay the contractual loan obligations. Interest for credit-risk assumption forms the earnings and rewards from such debt-obligations and risks.

The cash flow of the financer is impacted when the interest accrued and principal amounts are not paid. Further, the cost of collections also increases. Though, there is a grey area in guessing who and when will default on borrowings, it is the process of intelligent credit analysis that can help mitigate the severity of complete loss of the borrowings and its recovery.

Who Needs Credit-Risk Analysis?

Banks, financial institutions and NBFCs offer mortgages, loans, credit cards etc and need to exercise utmost caution in credit risk analysis. Similarly, companies that offer credit, bond issuers, insurance companies, and even investors need to know the techniques of effective risk analysis

Using Machine Learning in Credit Risk Analysis

With machine learning, banks and financial institutions are increasingly able to implement more science and less guesswork. Major financial institutions have been using AI to detect and prevent fraudulent transactions for several years.

AI-based scoring models combine customers' credit history and the power of big data, using a wider range of sources to improve credit decisions and often yielding better insights than a human analyst. Banks can analyse larger volumes of data – both financial and non-financial – by continuously running different combinations of variables and learning from that data to predict variable interactions.

We will use Predictive Modeling Techniques (using a **decision tree model),** which will predict the suspected credit card defaulters among the new applicants.

What is Predictive Modeling?

Predictive modeling is a process that uses data mining and probability to forecast outcomes. Each **model** is made up of a number of predictors, which are variables that are likely to influence future results. Once data has been collected for relevant predictors, a statistical **model** is formulated.

Specifically, some of the different types of predictive models are:

- Ordinary Least Squares.
- Generalized Linear Models (GLM)
- Logistic Regression.
- Random Forests.
- Decision Trees.
- Neural Networks.

We will build a custom decision tree model that will predict **the suspected credit card defaulters among the new applicants.**

Decision Tree

A **decision tree** is a **decision** support tool that uses a **tree**-like graph or model of **decisions** and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Advantages:

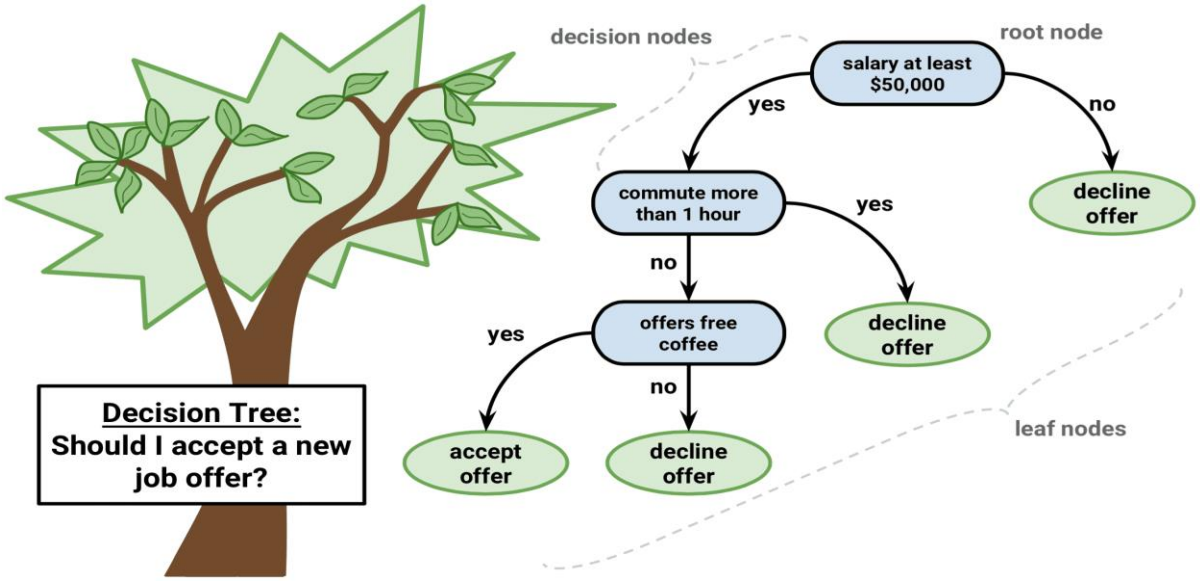
1. Compared to other algorithms decision trees requires less effort for data preparation during pre-processing.
2. A decision tree does not require normalization of data.
3. A decision tree does not require scaling of data as well.
4. Missing values in the data also does NOT affect the process of building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

Disadvantage:

1. A small change in the data can cause a large change in the structure of the decision tree causing instability.
2. For a Decision tree sometimes calculation can go far more complex compared to other algorithms.

3. Decision tree often involves higher time to train the model.
4. Decision tree training is relatively expensive as complexity and time taken is more.
5. Decision Tree algorithm is inadequate for applying regression and predicting continuous values.

Example:



Discribtion of Dataset:

age	gender	education	occupatio	organizatic	seniority	annual_inc	disposable	house_typ	vehicle_ty	marital_st	no_card	default
19	Male	Graduate	Profession	None	None	186319	21625	Family	None	Married	0	1
18	Male	Under Gra	Profession	None	None	277022	20442	Rented	None	Married	0	1
29	Male	Under Gra	Salaried	None	Entry	348676	24404	Rented	None	Married	1	1
18	Male	Graduate	Student	None	None	165041	2533	Rented	None	Married	0	1
26	Male	Post Gradu	Salaried	None	Mid-level	348745	19321	Rented	None	Married	1	1
26	Female	Other	Student	None	None	404972	22861	Family	None	Single	0	1
28	Male	Under Gra	Student	None	None	231185	20464	Family	None	Married	0	1
24	Female	Under Gra	Salaried	None	Entry	102554	42159	Family	None	Married	1	1
26	Female	Under Gra	Salaried	None	Junior	226786	19817	Family	None	Single	0	1
26	Male	Graduate	Salaried	None	Mid-level	250424	5271	Family	Two Whee	Married	1	1

This data set includes the data of credit card applicants along with the classification of whether they are defaulter or not. This dataset consists of following columns-

age	Age of the applicant
gender	Gender of applicant
education	Graduate Post Graduate Other Under Graduate
occupation	Professional Salaried Student Professional Business
organization_type	None Tier 3
seniority	Entry Junior Mid-level 1 None
annual_income	Annual income of the applicant like 186319 ,277022
house_type	Family Rented
disposable_income	Like 20442,24404
vehicle_type	None Two Wheeler
marital_status	Married Single
no_card	0 1 2
default	0 1

Now,heading towards the coding part.

Python Modules Used

- **Numpy - NumPy** is a **Python** package which stands for 'Numerical **Python**'. It is the core library for scientific computing, which contains a powerful n-dimensional array object. It is also useful in linear algebra, random number capability etc.
- **Pandas- Pandas** is an opensource library that allows to you perform data manipulation in **Python**. **Pandas** library is built on top of Numpy, meaning **Pandas** needs Numpy to operate. **Pandas** provide an easy way to create, manipulate and wrangle the data. **Pandas** is also an elegant solution for time series data.
- **Matplotlib- Matplotlib** is a plotting library for the **Python** programming language and its numerical mathematics extension NumPy.
- **Random-**
- **Sklearn (scikit learn)-**

  Scikit-learn is a free machine learning library for **Python**. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports **Python** numerical and scientific libraries like NumPy and SciPy .

DecisionTreeClassifier()

Parameers of dEcisionTreeClassifier are:

- criterion : string, optional (default="gini"). The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

Gini Index vs Entropy

Both gini and entropy are measures of impurity of a node.

Information gain is the entropy of parent node minus sum of weighted entropies of child nodes.
Weight of a child node is number of samples in the node/total samples of all child nodes. Similarly information gain is calculated with gini score,

$$Gini = 1 - \sum_{i=1}^{n} p^2(c_i)$$

$$Entropy = \sum_{i=1}^{n} -p(c_i)log_2(p(c_i))$$

where $p(c_i)$ is the probability/percentage of class c_i in a node.

- splitter : string, optional (default="best"). The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

- max_depth : int or None, optional (default=None). The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
-
- min_samples_split : int, float, optional (default=2). The minimum number of samples required to split an internal node:
- If int, then consider min_samples_split as the minimum number.
- If float, then min_samples_split is a fraction and ceil(min_samples_split * n_samples) are the minimum number of samples for each split.
-
- min_samples_leaf : int, float, optional (default=1). The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.
- If int, then consider min_samples_leaf as the minimum number.
- If float, then min_samples_leaf is a fraction and ceil(min_samples_leaf * n_samples) are the minimum number of samples for each node.
-
- min_weight_fraction_leaf : float, optional (default=0.). The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.
-
- max_features : int, float, string or None, optional (default=None). The number of features to consider when looking for the best split:
- If int, then consider max_features features at each split.
- If float, then max_features is a fraction and int(max_features * n_features) features are considered at each split.
- If “auto”, then max_features=sqrt(n_features).
- If “sqrt”, then max_features=sqrt(n_features).
- If “log2”, then max_features=log2(n_features).
- If None, then max_features=n_features.
-
- random_state : int, RandomState instance or None, optional (default=None). If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.
-
- max_leaf_nodes : int or None, optional (default=None). Grow a tree with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.
-
- min_impurity_decrease : float, optional (default=0.). A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
-
- min_impurity_split : float, (default=1e-7). Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf. Use min_impurity_decrease instead.
-
- class_weight : dict, list of dicts, “balanced” or None, default=None. Weights associated with classes in the form {class_label: weight}.

- presort : bool, optional (default=False). Whether to presort the data to speed up the finding of best splits in fitting. For the default settings of a decision tree on large datasets, setting this to true may slow down the training process. When using either a smaller dataset or a restricted depth, this may speed up the training.

train_test_split()

Split arrays or matrices into random **train** and **test** subsets

X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3, random_state=1234)

This will split data into two groups in which train dataset contains 70% of data and test dataset contains remaining 30%.

accuracy_score()

In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must *exactly* match the corresponding set of labels in y_true.

For visualizing a decision tree:

pydotplus- PyDotPlus is an improved version of the old pydot project that provides a **Python** Interface to Graphviz's Dot language.

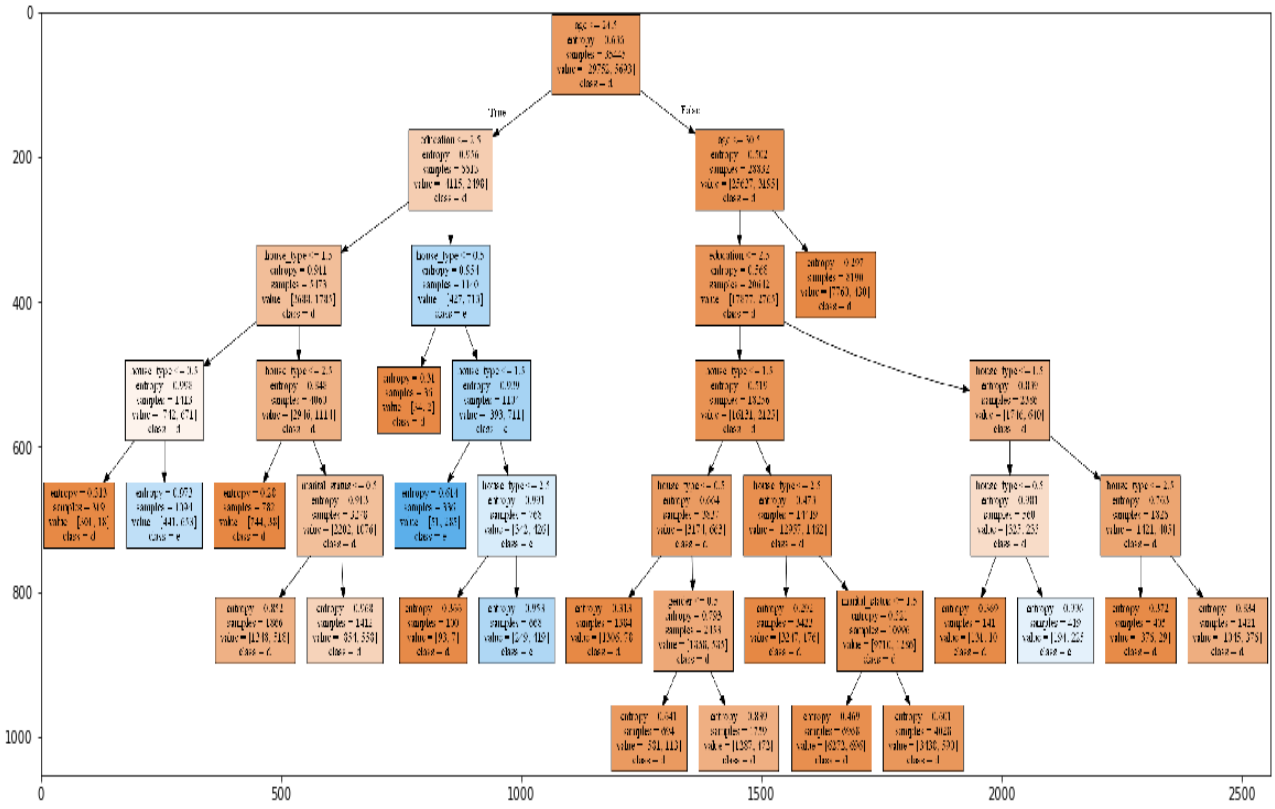
Graphviz- graphviz provides a simple pure-**Python** interface for the **Graphviz** graph-drawing software.

IPython - IPython (Interactive **Python**) is a command shell for interactive computing in multiple programming languages, originally developed for the **Python** programming language, that offers introspection, rich media, shell syntax, tab completion, and history.

Io- Core tools for working with streams. The **io** module provides the **Python** interfaces to stream handling.

Sys- The **sys** module is included in the standard libraries and contains the functions and other data necessary for your code to perform introspection about the system in which its running.

Output Dicision Tree



We got accuracy of **78.52%** via gini criterion and accuracy of **86.14%** by implementing Entropy criterion.

THANK YOU!!!