# Loan Approval using Machine Learning

Group 38

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

## Motivation

- In the era of digitalisation, many people are applying for loans.
- To see whether an individual is eligible for getting a loan by seeing his record manually is a tough task considering the fact that there are thousands of applications coming for loans everyday.
- Hence we need to find a digital solution which will help in predicting whether an individual is applicable for loan or not.

## Solution

- To solve the problem stated above, we propose Machine learning model which will predict whether an individual is applicable for loans or not.
- When an individual will send his/her details and bank account details to apply for a loan. The model will analyse the details of applicants and tell whether the applicant should be given loan or not.

## Approach

- We will first use preprocessing techniques on our data and then train different models such as: Decision Tree, K-Nearest Neighbour,  RandomForest, Logistic Regression Classifier.
- We are using GridSearchCV and RandomizedSearchCV for hypertuning the parameters.
- After finding the best parameters, we have used the ensemble approaches such as bagging classifier for better results.

## Data Analysis

- This is the dataset we got  from kaggle.
- The dataset consist of two files: application.csv and credit.csv
- Application.csv contains unique 438510 rows and 18 columns which contains personal details of an individual such as NumberOfChildren, Gender, Own_a_car.
- Credit.csv contains unique  1048576 rows and 3 columns which contains banking details of an individual such as the number of months loan have not been paid.
- The evaluation metrics used are:
    1. ROC Curve
    2. Confusion Matrix plot, f1-score, Accuracy, precision, recall
    3. Pairwise plot for correlation for feature selection.
    4. Train-test error vs parameter plot for different models.

## Summary of Main results

| Model Name | Precision | Recall | F1-score | Accuracy | AUC |
|---|---|---|---|---|---|
| Logistic Regression | 0.57 | 0.57 | 0.57 | 0.5667 | 0.576 |
| K-Nearest Neighbour | 0.75 | 0.73 | 0.73 | 0.7321 | 0.869 |
| Decision Tree Classifier | 0.82 | 0.81 | 0.81 | 0.8110 | 0.902 |
| Random Forest Classifier | **0.86** | **0.84** | **0.84** | **0.8430** | **0.911** |

## Summary of Main results

Best parameter for every model used are:-

Logistic Regression:-LogisticRegression(penalty='l2', n_jobs = -1, max_iter = 50)

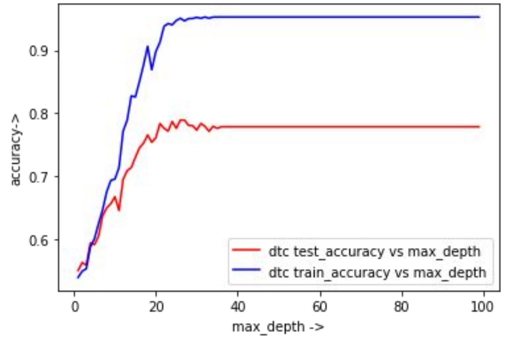
Random Forest:- RandomForestClassifier(bootstrap = False, n_estimators = 53, max_features='auto', random_state = 84, n_jobs = -1)

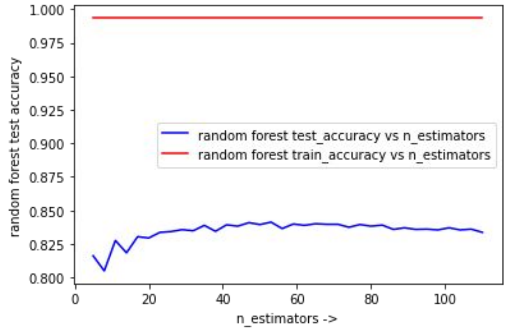
K-Nearest Neighbour:- KNeighborsClassifier(metric='manhattan',n_neighbors=98,weights='distance')

Decision Tree Classifier:- DecisionTreeClassifier(criterion="entropy", max_depth=26,max_features='auto',random_state=50)

## Analyses

**A model that is underfit will have high training and high testing error while an overfit model will have extremely low training error but a high testing error.**





Here the train and test accuracies are much higher than the base model. So, the model is not underfit.
Here, in left figure, the test accuracy increased till max_depth=26 after that it decrease and becomes constant. So, the model will be overfitted after max_depth=26 and same is the case in right figure at n_estimators=53. So, we took max_depth=26 and n_estimators=53 to avoid overfitting.

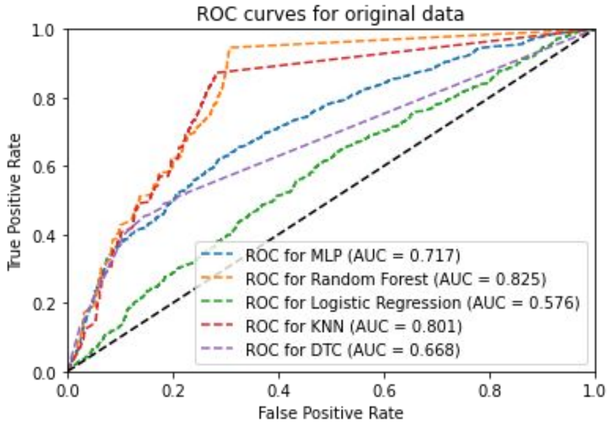
## Analysis

Steps for better accuracy than baseline accuracy.

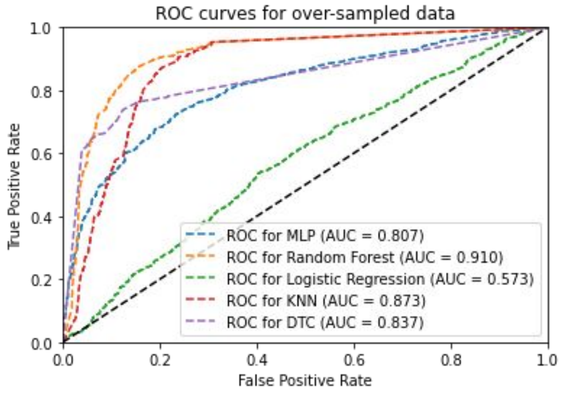
1. We dropped the columns with very high correlation. We dropped the column with many null values.
2. We did one hot encoding because some the data in columns is non-numerical.
3. We have done **over-sampling** of our data because our labels were not equally distributed hence for better results, we did oversampling of our data.
4. After applying over-sampling, **we did hyper-parameter tuning** for finding out the best parameters for our models. We did it using GridSearchCV and RandomizedSearchCV.
5. For better results, we applying ensemble approaches such as **Bagging Classifier** as ensemble for better performance.

# Plots after Analysis

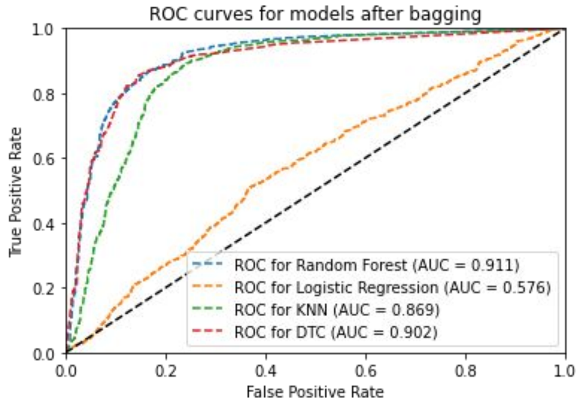
ROC curves on original data



ROC Curve for Oversampled Data



ROC curves for bagging models



## Conclusion and learnings

- Oversampling increased the precision, recall and f1-score of every model quite significantly.
- Hyperparameter tuning increases the accuracies by 1-2%.
- Ensemble approaches such as bagging increased the performance of our models to al little extent.
- Random forest with oversampling and bagging is our  best model, it provided best performance. It had accuracy of 84.3% and AUC 0.911.