Question 5: Detailed Description in steps

Step 1: Remove rows having current as loan status and define labels as described

Step 2: Remove columns where all the values are NAN.

Step 3:Remove columns having all unique values or all columns having same values

This combined brings down the dimension to `[24301 rows x 42 columns]`

Step 4:

- Dropping columns with low information and majority NAN columns contains only 0.37% or 0.42% 0.008% of data other values are NAN hence dropping these columns

Step 5:

- Remove emp_title, desc, title as it contains text that can be dropped. Removing grade and zip_code and keeping addr_state sub_grade instead dropping date columns

Step 6: Find and fill missing values:

- Emp_length filled with 0 years for entries which are blank.
- pub_rec_bankruptcies had only 3 values 0,1,2 hence filled null with a random number as it might indicate some order if 3 or -1 was used.
- revol_util is replaced by the median as the distribution was not symmetric and for asymmetric distributions median is a better choice. Decision of symmetric is taken by measuring skewness.

Step 7:

- Assign numeric value to each category value and use Label Encoder to encode them and then use label binarizer to convert such features into binary features revol_util that is a percentage is converted to a float similary int_rate is converted to float type.

Step 8:

- Employ feature selection using SelecKBest where k=’all’ and function score is chi2 score for all the features are calculated and features having scores below 10 are removed.

Step 9: Apply Gradient Boosting Clasifier.

Step 10: Tune hyper parameters for best accuracy calculate metric precision and recall

Step 11: Study effect of changing num_trees in the classifier.

Step 12: Compare best performance with decision tree.

(a) Pre-process the data as needed to apply the classifier to the training data (you are free to use pandas or other relevant libraries. Note that test data should not be used for pre-processing in any way, but the same pre-processing steps can be used on test data. Some steps to consider:

- Check for missing values, and how you want to handle them (you can delete the records, or replace the missing value with mean/median of the attribute - this is a decision you must make. Please document your decisions/choices in the final submitted report.)

  Step 6 describes the features with missing values and how they were handled

- Check whether you really need all the provided attributes, and choose the necessary attributes. (You can employ feature selection methods, if you are familiar; if not, you can eyeball.)
  - ***Some features were manually dropped. Step 5 involves features removed and the intuition behind it. Step 3 and Step 4 removes features that are as features having same value through or distinct value throughout are less likely to be used for making decision.***
  - ***Step 8 describes the method used and the threshold considered for feature selection***
- Transform categorical data into binary features, and any other relevant columns to suitable datatypes

  Step 7 converts categorical data to binary features.
- Any other steps that help you perform better

  Step 7,3,4,8 are steps that tries to improve the efficiency

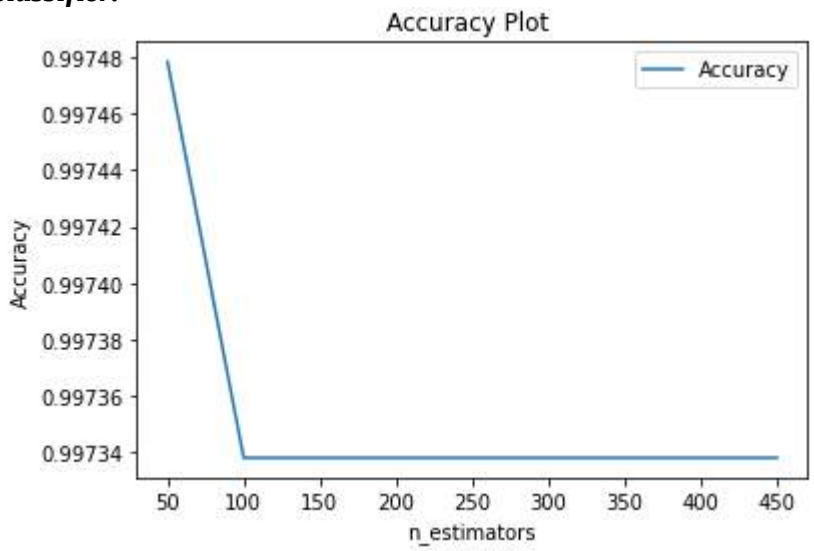
(b) Apply gradient boosting using the function sklearn.ensemble.GradientBoostingClassifier for training the model. You will need to import sklearn, sklearn.ensemble, and numpy. Your effort will be focused on predicting whether or not a loan is likely to default.

- Get the best test accuracy you can, and show what hyper-parameters led to this accuracy. Report the precision and recall for each of the models that you built.

  Hyperparameters for n_estimators=50 , learning_rate=0.4,max_depth=5 ,random_state=0

```
Precision
0.9980375665825624
Recall
0.9994385176866929
Accuracy
0.9974782852339591
```

• In particular, study the effect of increasing the number of trees in the classifier.



Accuracy normally increases as we increase number of trees but after a certain value i.e. 100 the accuracy does not increase but does not deteriorate either hence we can set the start point as the accuracy is same for 400 trees and 100 trees in this case the accuracy is slightly more (0.00014 i.e. 0.014%) for 50 hence 50 is taken as a hyper parameter.

• Compare your final best performance (accuracy, precision, recall) against a simple decision tree built using information gain. (You can use sklearn's inbuilt decision tree function for this.)

	DecisionTreeClassifier	GradientBoostngClassifier
Precision	0.9982	0.9980
Recall	0.9989	0.9994
Accuracy	99.72%	99.75%

As we can GradientBoostingClassifier is giving slightly better accuracy after tuning hyper parameters also recall is improved precision is less but the values are very close.

```
classifier2=GradientBoostingClassifier(n_estimators=50, learning_rate=0.4,
         max_depth=5, random_state=0).fit(X_train, y_train)
y_pred = classifier2.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
true_pos = np.sum(np.diag(cm))
false_pos=cm[0,1]
false_neg= cm[1,0]
precision = true_pos/(true_pos+false_pos)
recall =true_pos/(true_pos+false_neg)
print("Precision")
print(precision)
print("Recall")
print(recall)
print("Accuracy")
print(accuracy_score(y_test, y_pred))
```

```
Precision
0.9980375665825624
Recall
0.9994385176866929
Accuracy
0.9974782852339591
```

```
from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
true_pos = np.sum(np.diag(cm))
false_pos=cm[0,1]
false_neg= cm[1,0]
precision = true_pos/(true_pos+false_pos)
recall =true_pos/(true_pos+false_neg)
print("Precision")
print(precision)
print("Recall")
print(recall)
print("Accuracy")
print(accuracy_score(y_test, y_pred))
```

```
Precision
0.9982469672533483
Recall
0.998947442284752
Accuracy
0.997198094704399
```