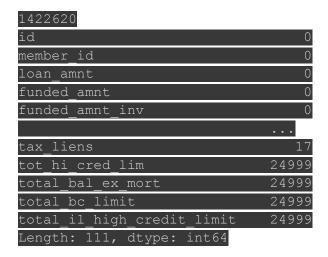
Answer 5

a)

Preprocessing: Using pandas we can preprocess raw training data suitable for applying ML algorithms.

Check for null values in the training data column-wise:



Drop columns with a significant percentage of missing values:

```
'desc': number of missing values '7943' ==> '32.686%'

'mths_since_last_delinq': number of missing values '15767' ==> '64.882%'
'revol_util': number of missing values '29' ==> '0.119%'
'collections_12_mths_ex_med': number of missing values '29' ==> '0.119%'
'chargeoff_within_12_mths': number of missing values '29' ==> '0.119%'
'pub_rec_bankruptcies': number of missing values '417' ==> '1.716%'
'tax_liens': number of missing values '17' ==> '0.070%'
```

- Drop columns with 90% null data are:

- Drop useless columns - Date columns, titles (redundant in purpose column), zipcode (encrypted), policy code (single unique value)

```
'last_pymnt_d','last_credit_pull_d','earliest_cr_line',
'url','emp_title','title','zip_code','policy_code', 'emp_length'
```

- Drop grade because it is just a sub-feature of sub_grade
- Drop issue date column as it is uncertain whether a loan will be issued for our problem statement
- Drop desc as not relevant info can be obtained from it
- Drop single unique value for columns initial list status and application type
- Remove all columns with single values whose mean, std, min, max all are 0.0 for columns collections_12_mths_ex_med, chargeoff_within_12_mths, tax_liens, deling_amnt, acc_now_deling, out_prncp, and out_prncp_inv
- To transform categorical data into binary features we create dummies for columns sub_grade, home_ownership, verification_status, purpose, addr_state which are object type and category
- Drop columns id and member id because those are irrelevant to loan status
- Check and remove duplicates columns & features
- Remap column term into {' 36 months': 36, ' 60 months': 60} and loan status into {'Fully Paid':1, 'Charged Off': -1}
- Replace NaN with respective mean of the column revol_util and pub rec bankruptcies
- Remove special characters like % from data present in columns int_rate and revol_util
- b) Build multiple GradientBoostingClassifier models by tweaking the hyperparameters and evaluating for best test accuracy. Hyperparameters considered for improving performance are learning rate, n estimators, and max features.

- The best test accuracy is obtained by the Gradient Boosting classifier:

Classifier:

Max Features: None Number of Trees: 150 Learning Rate: 0.2

Output:

Let us study the effect of increasing the number of trees in the classifier

Gradient Boosting Classifier	Precision, Recall, Accuracy of Test Accuracy
Max Features: None	Precision: 0.9794780641512483
Number of Trees: 20	Recall: 1.0
Learning Rate: 0.1	Accuracy: 98.22%
Max Features: None	Precision: 0.9883417577042231
Number of Trees: 50	Recall: 1.0
Learning Rate: 0.1	Accuracy: 99.00%
Max Features: None	Precision: 0.9925495333224169
Number of Trees: 75	Recall: 1.0
Learning Rate: 0.1	Accuracy: 99.36%
Max Features: None	Precision: 0.9940949725252194
Number of Trees: 100	Recall: 0.9998350243339107
Learning Rate: 0.1	Accuracy: 99.48%
Max Features: None	Precision: 0.9943396226415094
Number of Trees: 125	Recall: 0.9998350243339107
Learning Rate: 0.1	Accuracy: 99.50%
Max Features: None	Precision: 0.9945027896291434
Number of Trees: 150	Recall: 0.9998350243339107
Learning Rate: 0.1	Accuracy: 99.52%
Max Features: None	Precision: 0.9951559934318555
Number of Trees: 200	Recall: 0.9998350243339107
Learning Rate: 0.1	Accuracy: 99.57%

Max Features: None	Precision: 0.9958104000657192
Number of Trees: 300	Recall: 0.9999175121669553
Learning Rate: 0.1	Accuracy: 99.64%
Max Features: None	Precision: 0.9965474722564734
Number of Trees: 500	Recall: 1.0
Learning Rate: 0.1	Accuracy: 99.71%
Max Features: log2	Precision: 0.9906182987848463
Number of Trees: 150	Recall: 0.9145426049657676
Learning Rate: 0.5	Accuracy: 92.01%

- Let's see how Test_Accuracy varies on #Trees in Gradient Boosting Classifier

Max Features: None Learning Rate: 0.1

	Test_Accuracy(%)
#ClassifierTrees	
20	98.22
50	99.00
75	99.36
100	99.48
125	99.50
150	99.52
200	99.57
300	99.64
500	99.71

- Comparing the best Gradient Boosting classifier Vs Random Forest Classifier:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None, criterion='gini', max_depth=None, max_features='auto', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None, oob_score=False, random_state=None, verbose=0, warm_start=False)
```

Output:
Test Result:
Precision : 0.9910853030179112
Recall : 0.9995875608347768
Accuracy: 99.20%