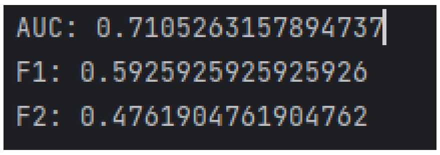
1. Preprocessing Data for Loan Default Prediction
    (a) F_1 score is the harmonic mean of precision and recall, while F_2 score gives the recall more weights than the precision.
    Since we are predicting whether an applicant would default on the loan, we would want to pay more attention to reduce the missing identification of those who would default (false negative) rather than to reduce misclassifying a reliable applicant as someone who would default (false positive), otherwise, we would end up lending money to someone who will default, resulting in financial losses.
    Since the recall reflects our ability to identify someone who would default (true positive) among all the applicants that would default (true positive + false negative), and F_2 score gives more weights to recall, optimizing F_2 would be preferred over F_1 in this situation.
    (b) I would randomly partition the data with a train: test sets ratio of 80%: 20%. A train size of 80% ensures that the majority of the data is used for training to let the model capture the underlying patterns as much as possible. A test size of 20% is reserved to evaluate the final trained model's generalizability on unseen data. The test set should have a significant portion of the total samples to make sure the performances of the model tested on them are representative enough. Since we are using GridSearchCV in the next question, which would perform a 5-fold cross validation by default, I would not specifically split a validation set here. The partition code is partition_1b() in test.py.
    (f) I preprocessed the loan_default.csv dataset to transfer all categorical data into numerical data (preprocessing() in test.py). For compute_correlation(), I tried all three methods (Pearson, Spearman, and Kendall), and chose to discard one feature from any pair of features with correlation score higher than 0.8. By removing highly correlated data, I can reduce the redundancy of the data, and improve the generalization of the model. All three methods end up removing the same features: column 4 and 5.
    For rank_correlation() and rank_mutual(), I choose to include the top 10 features that has the highest absolute correlation criteria or mutual information with the class labels and discard the rest. Removing the features that has lower correlation with the label helps ensure that the model would capture the key relation between the features and the label and reduce model's confusion on irrelevant features. The top 10 features for the two methods are as follows:
    Rank_correlation(): [19, 3, 5, 11, 2, 8, 9, 17, 25, 24]
    Rank_mutual(): [19, 5, 3, 25, 18, 13, 11, 7, 24, 6]
2. Decision Tree to Predict Loan Defaults
    (b) The best choice is a best-depth of 4 and a best-leaf-samples of 100. Figure 2 shows the validation AUC scores of different combinations of max depth and min sample leaf.
    (c) Figure 3 shows the AUC, F1, and F2 scores on the test set of the decision tree model with the optimal parameters (as in 2(b)) trained on the entire training data.
    (d) Figure 1 shows the visualization of the top 3 levels of the decision tree from 2c.
    (e) Table 1 shows the performances of decision tree models trained on the entire training set. Each model uses different feature selection methods for fine tuning, and thus has different set of optimal parameters. For 1(c) (correlation matrix), since all Pearson, Kendall and Spearman correlations would drop the same features, it does not matter which correlation method we choose. Here, we are using Spearman correlation.



```
AUC: 0.7105263157894737
F1: 0.5925925925925926
F2: 0.4761904761904762
```

Figure 3 Test Performances of Decision Tree with Optimal Parameters

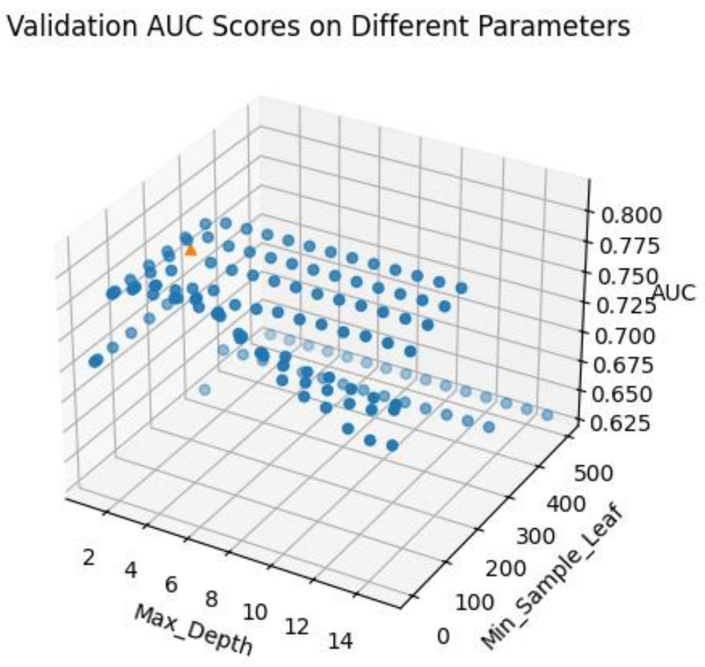


Figure 2 Validation AUC Scores on Different Parameters

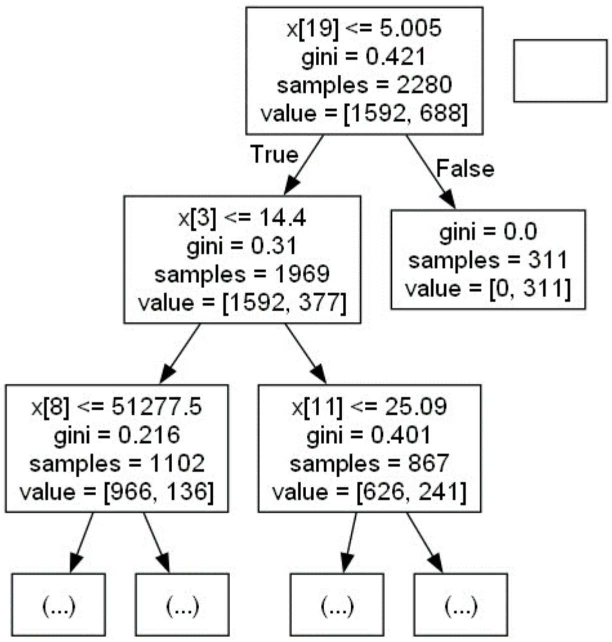


Figure 1 Visualization of Top 3 Levels of Decision Tree

(f) From Table 1, we can see that the correlation matrix of features leads to slightly

higher AUC, F1, and F2 scores compared to correlation criteria and mutual information. This could be because that correlation matrix only eliminates 2 features out of the 26 features, while the correlation criteria and mutual information only keep 10 features. However, the differences in performances are not significant.
As for correlation criteria and mutual information, they lead to decision tree models that has exactly the same performances. This could be because that the 10 features they select respectively are very similar (refer to 1(f)). Thus, they could yield similar optimal parameters, leading to similar decision tree models.

Table 1 Performances of Decision Tree on Entire Training Set using Different Feature Selection Methods for Finetuning

method	AUC	F1	F2
Correlation matrix	0.7092777048789216	0.5896414342629482	0.4939919893190921
Correlation Criteria	0.7076673028748658	0.5867768595041323	0.47972972972972966
Mutual Information	0.7076673028748658	0.5867768595041323	0.47972972972972966

3. Spam Detection via Perceptron
   (a) I plan to split the whole dataset into train: validation: test = 70%: 15%: 15%. A larger training set size would make sure that the model captures the underlying relations as much as possible during the validation process. The validation and test set shares the rest of the dataset equally. The test set should large enough to be representative of the real-world unseen data, and the validation set should be similar with test set to make sure the fine-tuning process is in the similar environment as which the model would finally be tested on. I did not use cross validation techniques because they could be costly in terms of computation. However, this means that my validation score could be

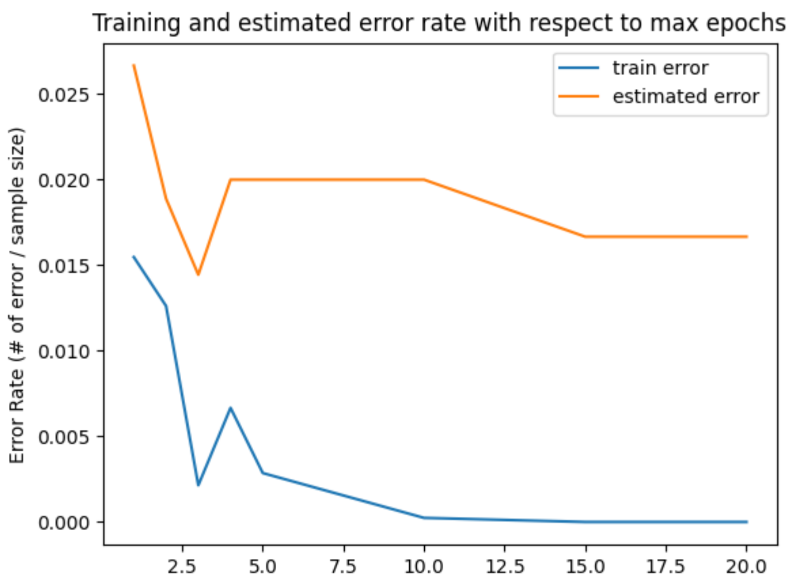


Figure 4 Training and estimated error rate with respect to max epochs

biased and would not be representative enough.

(g) Figure 4 shows the final training error and estimated generalization error rates of the perceptron with different max epochs. Based on Figure 4, the optimal parameter would be setting max epochs = 15. This is the point where estimated error rate stops decreasing. 393 mistakes are made before the algorithm terminates with classifying all training points properly.

(k) Figure 5 shows the final training error rate and the estimated generalization error rate of the average perceptron with different max epochs.

(l) Based on the performances from 3(g) and 3(k), the optimal algorithm is the perceptron with a max epoch of 5. The expected error rate is 0.037.

(m) The 15 words with the most positive weights: 'sendgreatoff', 'enemi', 'seem', 'collect', 'numberpm', 'forget', 'pleasur', 'method', 'present', 'jewelri', 'numberb', 'feet', 'six', 'mandat', 'thi'.

The 15 words with the most negative weights: 'that', 't', 'no', 'necessarili', 'at', 'dn', 'script', 'left', 'gr', 'servic', 'welcom', 'sai', 'precis', 'built', 'craft'.

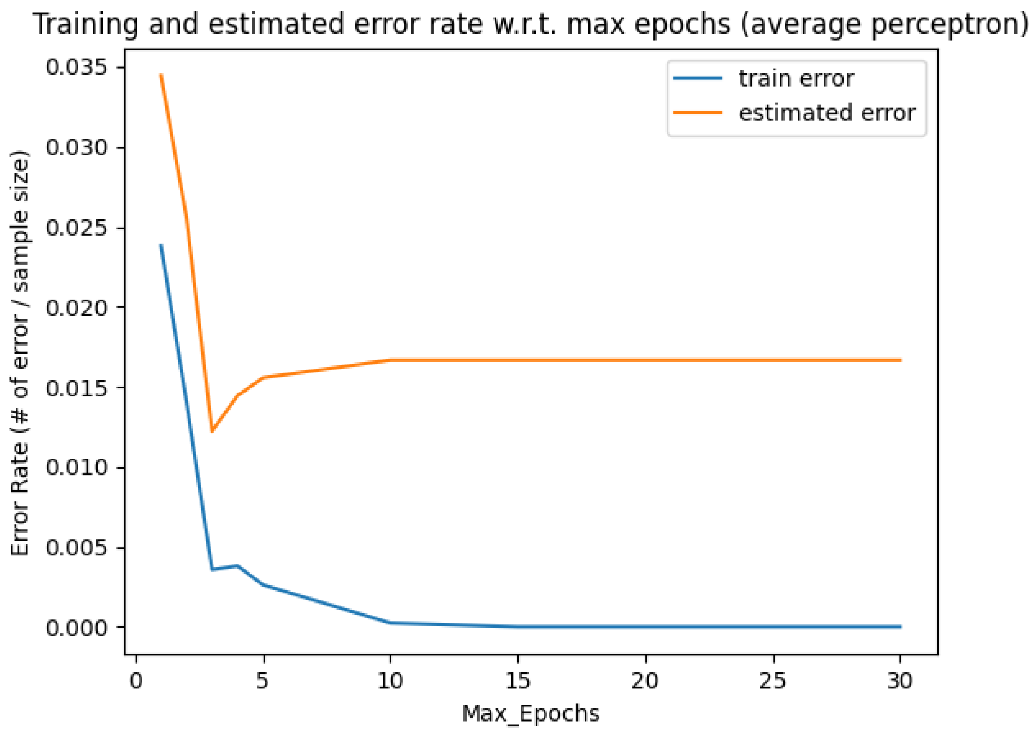


Figure 5 Training and estimated error rate w.r.t. max epochs (average perceptron)