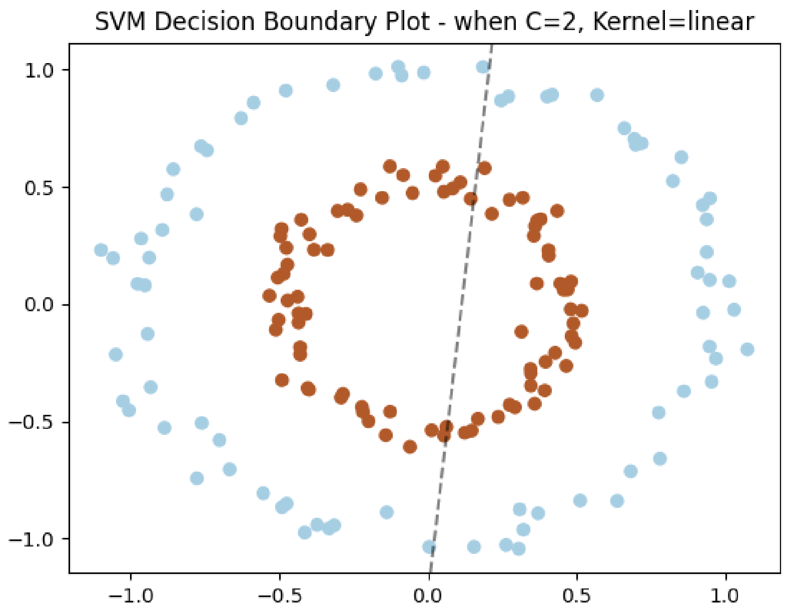
CENG 499 Homework-3 Report

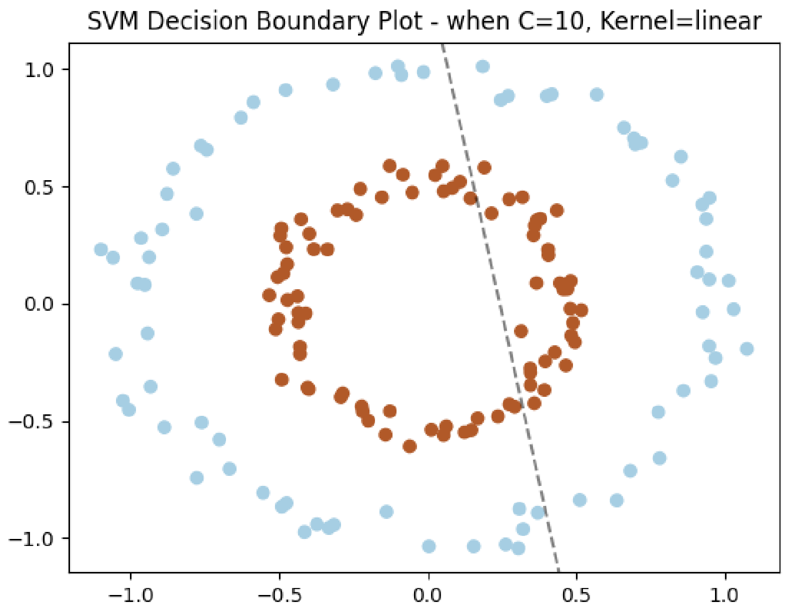
Ismail Karabas (2375186)

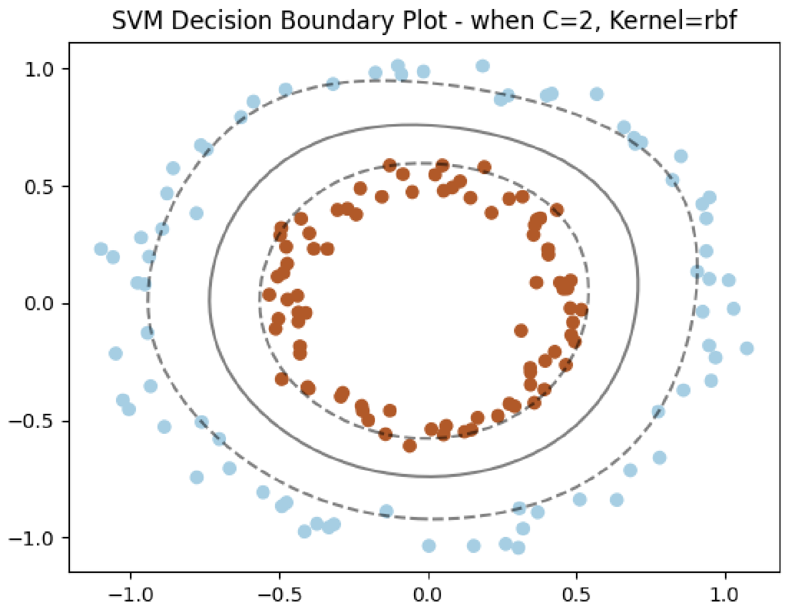
December 29, 2023

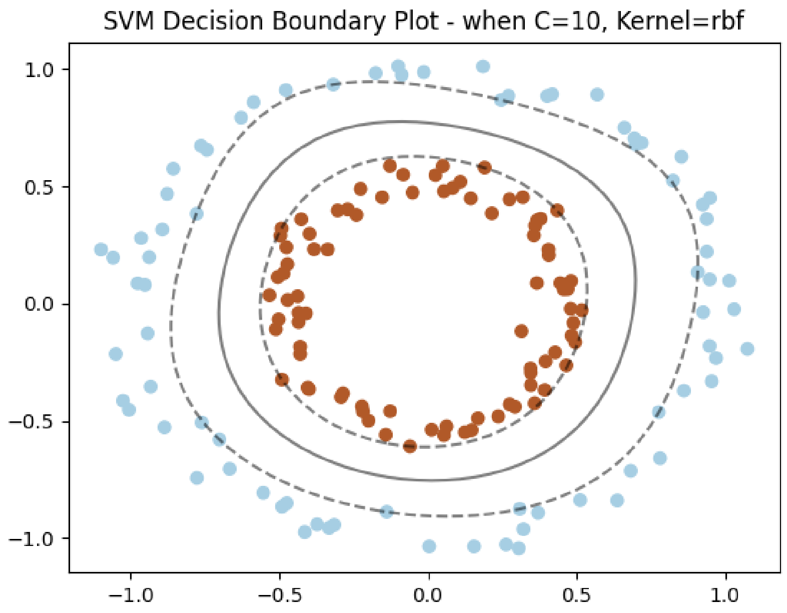
PART 2

1. **Plots for** `part2_dataset1`




SVM Decision Boundary Plot - when C=10, Kernel=linear
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0


SVM Decision Boundary Plot - when C=2, Kernel=rbf
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0



The plots were consistent with the properties of reqularization parameter which are as follows:

When C is small, the SVM allows for a larger margin and more training errors. The model is more tolerant of misclassified training examples and focuses on finding a decision boundary that generalizes well to unseen data. This corresponds to a "softer" margin.

When C is large, the SVM aims to classify all training examples correctly. It prioritizes achieving a lower training error, potentially leading to a smaller margin and increased sensitivity to outliers or noise. This corresponds to a "harder" margin.

2. **Cross-validation to find the best hyperparameter values on** `part2_dataset2`

   The results are as follows :

```
Configuration: Regularization parameter = 0.1 | Kernel = linear || Accuracy = 0.8773333
Configuration: Regularization parameter = 1 | Kernel = linear || Accuracy = 0.957333333
Configuration: Regularization parameter = 10 | Kernel = linear || Accuracy = 0.95333333
Configuration: Regularization parameter = 100 | Kernel = linear || Accuracy = 0.9466666
Configuration: Regularization parameter = 0.1 | Kernel = rbf || Accuracy = 0.828000000
```

```
  Configuration: Regularization parameter = 1 | Kernel = rbf || Accuracy = 0.922666666666
  Configuration: Regularization parameter = 10 | Kernel = rbf || Accuracy = 0.93733333333
  Configuration: Regularization parameter = 100 | Kernel = rbf || Accuracy = 0.9146666666

  Best Configuration: Regularization parameter = 1 | Kernel = linear || Accuracy = 0.9573
```

PART 2

THE OVERALL EXPERIMENT SETUP

The implementation for all three questions can be found in the part3.py file. For the first question, I established two cross-validation loops—one for the outer loop and the other for the inner loop. At the beginning of each iteration of the outer loop, I applied MinMaxScaler as specified in the homework description. I ensured consistency across all algorithms by using the same preprocessed data for each inner loop cross-validation.

To facilitate a fair performance comparison between algorithms, I repeated the inner loop for the Random Forest algorithm five times and computed averages to mitigate the impact of its random initialization. Following each iteration of the outer loop, the results were stored in dedicated data structures for each algorithm.

Upon completing the cross-validation process, I utilized the stored experiment results to print the outcomes and their corresponding confidence intervals. This entire process was iterated twice, each time employing a different scoring type for use in the cross-validation comparisons.

Experiment results of nested cross-validation

```
#### scoring used for inner cross validation: F1 ####

### KNN Performance ###

Hyperparameter Configuration F1 Accuracy
--------------------------------------------------
1. {'metric': 'euclidean', 'n_neighbors': 5} 0.6731 0.7006
2. {'metric': 'euclidean', 'n_neighbors': 10} 0.7041 0.7177
3. {'metric': 'manhattan', 'n_neighbors': 5} 0.7179 0.7327
4. {'metric': 'euclidean', 'n_neighbors': 5} 0.6808 0.6976
5. {'metric': 'manhattan', 'n_neighbors': 10} 0.7265 0.7357
6. {'metric': 'manhattan', 'n_neighbors': 5} 0.7081 0.7357
7. {'metric': 'manhattan', 'n_neighbors': 5} 0.7224 0.7395
8. {'metric': 'euclidean', 'n_neighbors': 10} 0.6890 0.7027
9. {'metric': 'euclidean', 'n_neighbors': 10} 0.6758 0.6997
10. {'metric': 'manhattan', 'n_neighbors': 10} 0.7175 0.7305
11. {'metric': 'manhattan', 'n_neighbors': 5} 0.6895 0.7147
12. {'metric': 'manhattan', 'n_neighbors': 5} 0.6835 0.7117
13. {'metric': 'euclidean', 'n_neighbors': 10} 0.7049 0.7186
14. {'metric': 'manhattan', 'n_neighbors': 5} 0.6973 0.7267
```

```
15. {'metric': 'manhattan', 'n_neighbors': 5} 0.7255 0.7387

Average F1: 0.7011
Average Accuracy: 0.7202
Confidence Interval (F1): [0.6740666655787807, 0.7261466799242646]
Confidence Interval (Accuracy): [0.69833800866735, 0.7392471813130496]

==================================================
### SVM Performance ###

Hyperparameter Configuration F1 Accuracy
--------------------------------------------------
1. {'C': 0.5, 'kernel': 'rbf'} 0.6487 0.7186
2. {'C': 0.1, 'kernel': 'linear'} 0.7327 0.7538
3. {'C': 0.1, 'kernel': 'linear'} 0.7277 0.7477
4. {'C': 0.1, 'kernel': 'linear'} 0.6925 0.7305
5. {'C': 0.1, 'kernel': 'linear'} 0.7137 0.7327
6. {'C': 0.5, 'kernel': 'linear'} 0.7458 0.7598
7. {'C': 0.1, 'kernel': 'linear'} 0.7396 0.7695
8. {'C': 0.1, 'kernel': 'linear'} 0.7123 0.7387
9. {'C': 0.5, 'kernel': 'linear'} 0.7235 0.7297
10. {'C': 0.1, 'kernel': 'linear'} 0.7385 0.7605
11. {'C': 0.5, 'kernel': 'rbf'} 0.6616 0.7327
12. {'C': 0.1, 'kernel': 'linear'} 0.7229 0.7387
13. {'C': 0.1, 'kernel': 'linear'} 0.7466 0.7635
14. {'C': 0.1, 'kernel': 'linear'} 0.7222 0.7508
15. {'C': 0.5, 'kernel': 'linear'} 0.7123 0.7177

Average F1: 0.7160
Average Accuracy: 0.7430
Confidence Interval (F1): [0.653199841136236, 0.7463399528431848]
Confidence Interval (Accuracy): [0.7180135225045405, 0.7673652694610779]

==================================================
### Decision Tree Performance ###

Hyperparameter Configuration F1 Accuracy
--------------------------------------------------
1. {'max_depth': 10} 0.6844 0.6826
2. {'max_depth': 10} 0.6818 0.6907
3. {'max_depth': 10} 0.6755 0.6817
4. {'max_depth': 10} 0.6677 0.6707
5. {'max_depth': 10} 0.6739 0.6697
6. {'max_depth': 20} 0.6819 0.6757
7. {'max_depth': 10} 0.6642 0.6826
8. {'max_depth': 10} 0.6491 0.6486
```

```
9. {'max_depth': 10} 0.6669 0.6727
10. {'max_depth': 10} 0.6648 0.6617
11. {'max_depth': 10} 0.6645 0.6727
12. {'max_depth': 10} 0.6646 0.6637
13. {'max_depth': 10} 0.6848 0.6886
14. {'max_depth': 10} 0.7195 0.7207
15. {'max_depth': None} 0.6417 0.6366

Average F1: 0.6724
Average Accuracy: 0.6746
Confidence Interval (F1): [0.6443160288626886, 0.7073301603554539]
Confidence Interval (Accuracy): [0.6408408408408408, 0.7102102102102101]

==================================================
### Random Forest Performance ###

Hyperparameter Configuration F1 Accuracy
--------------------------------------------------
1. {'max_depth': 10, 'n_estimators': 20} 0.7034 0.7275
2. {'max_depth': 10, 'n_estimators': 20} 0.6938 0.7297
3. {'max_depth': 10, 'n_estimators': 20} 0.7293 0.7447
4. {'max_depth': 10, 'n_estimators': 20} 0.7134 0.7515
5. {'max_depth': 10, 'n_estimators': 20} 0.7227 0.7417
6. {'max_depth': 10, 'n_estimators': 20} 0.7050 0.7387
7. {'max_depth': 10, 'n_estimators': 20} 0.7292 0.7635
8. {'max_depth': 10, 'n_estimators': 20} 0.7185 0.7508
9. {'max_depth': 10, 'n_estimators': 20} 0.7216 0.7387
10. {'max_depth': 10, 'n_estimators': 20} 0.7247 0.7515
11. {'max_depth': 10, 'n_estimators': 20} 0.6985 0.7357
12. {'max_depth': 10, 'n_estimators': 20} 0.7220 0.7598
13. {'max_depth': 10, 'n_estimators': 20} 0.7553 0.7784
14. {'max_depth': 10, 'n_estimators': 20} 0.7244 0.7628
15. {'max_depth': 10, 'n_estimators': 20} 0.7497 0.7658

Average F1: 0.7208
Average Accuracy: 0.7494
Confidence Interval (F1): [0.6954421178381971, 0.7533335177374745]
Confidence Interval (Accuracy): [0.7283095970221719, 0.7740060419701138]

==================================================
==================================================
==================================================
==================================================

#### scoring used for inner cross validation: accuracy ####
```

```
### KNN Performance ###

Hyperparameter Configuration F1 Accuracy
--------------------------------------------------
1. {'metric': 'manhattan', 'n_neighbors': 5} 0.6937 0.7156
2. {'metric': 'manhattan', 'n_neighbors': 10} 0.6911 0.7087
3. {'metric': 'manhattan', 'n_neighbors': 5} 0.7066 0.7297
4. {'metric': 'manhattan', 'n_neighbors': 5} 0.6817 0.7006
5. {'metric': 'manhattan', 'n_neighbors': 5} 0.7293 0.7447
6. {'metric': 'manhattan', 'n_neighbors': 10} 0.6753 0.7087
7. {'metric': 'euclidean', 'n_neighbors': 5} 0.7087 0.7216
8. {'metric': 'manhattan', 'n_neighbors': 5} 0.6864 0.7087
9. {'metric': 'euclidean', 'n_neighbors': 10} 0.7053 0.7147
10. {'metric': 'manhattan', 'n_neighbors': 5} 0.6761 0.6976
11. {'metric': 'manhattan', 'n_neighbors': 5} 0.7077 0.7237
12. {'metric': 'manhattan', 'n_neighbors': 5} 0.6847 0.7087
13. {'metric': 'manhattan', 'n_neighbors': 5} 0.6961 0.7186
14. {'metric': 'euclidean', 'n_neighbors': 10} 0.6904 0.6997
15. {'metric': 'manhattan', 'n_neighbors': 3} 0.6656 0.6847

Average F1: 0.6932
Average Accuracy: 0.7124
Confidence Interval (F1): [0.6689956730357794, 0.7220961870730983]
Confidence Interval (Accuracy): [0.6892067216917517, 0.7394894894894894]

==================================================
### SVM Performance ###

Hyperparameter Configuration F1 Accuracy
--------------------------------------------------
1. {'C': 0.1, 'kernel': 'linear'} 0.7363 0.7545
2. {'C': 0.1, 'kernel': 'linear'} 0.7239 0.7508
3. {'C': 0.5, 'kernel': 'linear'} 0.7227 0.7417
4. {'C': 0.5, 'kernel': 'linear'} 0.7505 0.7575
5. {'C': 0.1, 'kernel': 'linear'} 0.7588 0.7748
6. {'C': 0.5, 'kernel': 'linear'} 0.7407 0.7538
7. {'C': 0.1, 'kernel': 'linear'} 0.7354 0.7605
8. {'C': 0.1, 'kernel': 'linear'} 0.7413 0.7658
9. {'C': 0.1, 'kernel': 'linear'} 0.7058 0.7267
10. {'C': 0.1, 'kernel': 'linear'} 0.7374 0.7575
11. {'C': 0.1, 'kernel': 'linear'} 0.7246 0.7477
12. {'C': 0.5, 'kernel': 'linear'} 0.7266 0.7447
13. {'C': 0.5, 'kernel': 'linear'} 0.7360 0.7455
14. {'C': 0.1, 'kernel': 'linear'} 0.7378 0.7598
15. {'C': 0.5, 'kernel': 'linear'} 0.7066 0.7207
```

```
Average F1: 0.7323
Average Accuracy: 0.7508
Confidence Interval (F1): [0.7060464058807268, 0.7558551871766309]
Confidence Interval (Accuracy): [0.7228228228228228, 0.7716216216216216]

==================================================
### Decision Tree Performance ###

Hyperparameter Configuration F1 Accuracy
--------------------------------------------------
1. {'max_depth': None} 0.6912 0.6916
2. {'max_depth': 20} 0.6617 0.6637
3. {'max_depth': 10} 0.6868 0.6877
4. {'max_depth': None} 0.6496 0.6467
5. {'max_depth': 10} 0.6832 0.6787
6. {'max_depth': 10} 0.6687 0.6697
7. {'max_depth': None} 0.6492 0.6497
8. {'max_depth': 20} 0.6833 0.6847
9. {'max_depth': 10} 0.6806 0.6757
10. {'max_depth': 10} 0.6919 0.6886
11. {'max_depth': 20} 0.7212 0.7237
12. {'max_depth': 20} 0.6723 0.6697
13. {'max_depth': None} 0.6697 0.6707
14. {'max_depth': 10} 0.6876 0.7027
15. {'max_depth': 20} 0.6701 0.6637

Average F1: 0.6778
Average Accuracy: 0.6778
Confidence Interval (F1): [0.6493211702082322, 0.7109509148228635]
Confidence Interval (Accuracy): [0.647754491017964, 0.7163663663663664]

==================================================
### Random Forest Performance ###

Hyperparameter Configuration F1 Accuracy
--------------------------------------------------
1. {'max_depth': 10, 'n_estimators': 20} 0.7010 0.7246
2. {'max_depth': None, 'n_estimators': 20} 0.7277 0.7477
3. {'max_depth': 10, 'n_estimators': 20} 0.7388 0.7628
4. {'max_depth': 10, 'n_estimators': 20} 0.6737 0.6946
5. {'max_depth': 10, 'n_estimators': 20} 0.7508 0.7808
6. {'max_depth': 10, 'n_estimators': 20} 0.6802 0.7177
7. {'max_depth': None, 'n_estimators': 20} 0.7164 0.7395
8. {'max_depth': 10, 'n_estimators': 20} 0.7026 0.7357
9. {'max_depth': 10, 'n_estimators': 20} 0.7254 0.7417
10. {'max_depth': 10, 'n_estimators': 20} 0.7261 0.7575
```

```
11. {'max_depth': 10, 'n_estimators': 20} 0.7296 0.7538
12. {'max_depth': None, 'n_estimators': 20} 0.7116 0.7267
13. {'max_depth': 10, 'n_estimators': 20} 0.6939 0.7246
14. {'max_depth': 10, 'n_estimators': 20} 0.7262 0.7477
15. {'max_depth': None, 'n_estimators': 20} 0.7254 0.7357

Average F1: 0.7153
Average Accuracy: 0.7394
Confidence Interval (F1): [0.675950727158331, 0.7465741041628757]
Confidence Interval (Accuracy): [0.7026982071892252, 0.7744744744744745]

==================================================
```

Question 2

For question two, I compiled a list containing all the function names. Subsequently, I trained a decision tree using the provided data. Utilizing the trained model, I extracted the most important features. Following this, I arranged the features without losing their respective indices and printed the top five features based on their importance.

Output

```
Top 5 Most Important Features:
Age in years - Importance: 0.14595286520857528
Duration in month - Importance: 0.1281337213137864
Credit amount - Importance: 0.12792619545542347
Status of existing checking account - Importance: 0.11552592717659435
Savings account / bonds - Importance: 0.06156633744196838
```

Question 3

The following is the reversed output of positive and negative support vectors. There are so many distinguishing features but basically in terms of positive things like having more savings (in bank accounts) in positive class and less in negative class. And many more intuitive differences between the overall statistics of these two class

```
    [Running] python -u "c:\ceng4_1\499\Hws\hw3\HW 3 Data and Source Codes (1)\HW 3 Data and

Common values for Status of existing checking account in Positive Class:
Value 3.0: Count 151
Value 0.0: Count 68
Value 1.0: Count 71
Value 2.0: Count 22

Common values for Status of existing checking account in Negative Class:
Value 1.0: Count 100
Value 0.0: Count 134
Value 2.0: Count 14
Value 3.0: Count 46

Common values for Duration in month in Positive Class:
Value 12.0: Count 37
Value 42.0: Count 6
Value 36.0: Count 38
Value 30.0: Count 20
Value 6.0: Count 16
Value 18.0: Count 30
Value 10.0: Count 6
Value 9.0: Count 9
Value 24.0: Count 74
Value 27.0: Count 6
Value 11.0: Count 2
Value 54.0: Count 1
Value 15.0: Count 19
Value 48.0: Count 17
Value 60.0: Count 6
Value 21.0: Count 8
Value 47.0: Count 1
Value 13.0: Count 2
Value 22.0: Count 2
Value 39.0: Count 4
Value 28.0: Count 2
Value 20.0: Count 2
Value 33.0: Count 1
Value 26.0: Count 1
Value 4.0: Count 1
Value 45.0: Count 1

Common values for Duration in month in Negative Class:
Value 48.0: Count 27
Value 24.0: Count 56
Value 30.0: Count 13
Value 12.0: Count 49
Value 60.0: Count 4
Value 45.0: Count 4
Value 18.0: Count 42
Value 36.0: Count 36
Value 42.0: Count 3
Value 54.0: Count 1
Value 33.0: Count 1
Value 21.0: Count 9
Value 15.0: Count 12
Value 9.0: Count 14
Value 27.0: Count 5
Value 14.0: Count 1
Value 6.0: Count 7
Value 8.0: Count 1
Value 10.0: Count 3
Value 72.0: Count 1
Value 39.0: Count 1
Value 40.0: Count 1
Value 20.0: Count 1
Value 16.0: Count 1
Value 28.0: Count 1

Common values for Credit history in Positive Class:
Value 4.0: Count 102
Value 2.0: Count 154
Value 0.0: Count 8
Value 1.0: Count 13
Value 3.0: Count 35

Common values for Credit history in Negative Class:
Value 2.0: Count 165
Value 3.0: Count 28
Value 4.0: Count 50
Value 0.0: Count 24
Value 1.0: Count 27

Common values for Purpose in Positive Class:
Value 6.0: Count 13
Value 2.0: Count 49
Value 9.0: Count 34
Value 1.0: Count 40
Value 3.0: Count 104
Value 0.0: Count 53
Value 4.0: Count 6
Value 5.0: Count 6
Value 10.0: Count 5
Value 8.0: Count 2

Common values for Purpose in Negative Class:
Value 3.0: Count 62
Value 0.0: Count 85
Value 9.0: Count 33
Value 1.0: Count 17
Value 2.0: Count 58
Value 6.0: Count 22
Value 5.0: Count 8
Value 10.0: Count 4
Value 8.0: Count 1
Value 4.0: Count 4




Common values for Savings account / bonds in Positive Class:
Value 0.0: Count 172
Value 4.0: Count 67
Value 1.0: Count 30
Value 3.0: Count 20
Value 2.0: Count 23

Common values for Savings account / bonds in Negative Class:
Value 0.0: Count 213
Value 1.0: Count 33
Value 4.0: Count 31
Value 2.0: Count 11
Value 3.0: Count 6

Common values for Present employment since in Positive Class:
Value 3.0: Count 59
Value 2.0: Count 105
Value 1.0: Count 54
Value 4.0: Count 80
Value 0.0: Count 14

Common values for Present employment since in Negative Class:
Value 2.0: Count 103
Value 0.0: Count 22
Value 1.0: Count 70
Value 4.0: Count 61
Value 3.0: Count 38

Common values for Installment rate in percentage of disposable income in Positive Class:
Value 2.0: Count 73
Value 3.0: Count 51
Value 4.0: Count 156
Value 1.0: Count 32

Common values for Installment rate in percentage of disposable income in Negative Class:
Value 2.0: Count 61
Value 3.0: Count 44
Value 4.0: Count 157
Value 1.0: Count 32
```

```
Common values for Personal status and sex in Positive Class:
Value 2.0: Count 177
Value 3.0: Count 30
Value 1.0: Count 97
Value 0.0: Count 8

Common values for Personal status and sex in Negative Class:
Value 1.0: Count 108
Value 2.0: Count 141
Value 3.0: Count 25
Value 0.0: Count 20

Common values for Other debtors / guarantors in Positive Class:
Value 0.0: Count 284
Value 2.0: Count 19
Value 1.0: Count 9

Common values for Other debtors / guarantors in Negative Class:
Value 0.0: Count 266
Value 1.0: Count 18
Value 2.0: Count 10

Common values for Present residence since in Positive Class:
Value 3.0: Count 49
Value 4.0: Count 117
Value 2.0: Count 96
Value 1.0: Count 50

Common values for Present residence since in Negative Class:
Value 2.0: Count 93
Value 4.0: Count 122
Value 1.0: Count 36
Value 3.0: Count 43

Common values for Property in Positive Class:
Value 0.0: Count 83
Value 1.0: Count 85
Value 3.0: Count 33
Value 2.0: Count 111

Common values for Property in Negative Class:
Value 0.0: Count 60
Value 3.0: Count 63
Value 2.0: Count 101
Value 1.0: Count 70
```

```
Common values for Age in years in Positive Class:
Value 49.0: Count 3
Value 45.0: Count 5
Value 35.0: Count 14
Value 25.0: Count 15
Value 44.0: Count 4
Value 39.0: Count 5
Value 42.0: Count 7
Value 36.0: Count 12
Value 30.0: Count 18
Value 33.0: Count 8
Value 37.0: Count 8
Value 24.0: Count 17
Value 23.0: Count 17
Value 29.0: Count 11
Value 31.0: Count 10
Value 48.0: Count 5
Value 41.0: Count 4
Value 51.0: Count 1
Value 58.0: Count 3
Value 47.0: Count 4
Value 34.0: Count 6
Value 21.0: Count 5
Value 27.0: Count 18
Value 28.0: Count 13
Value 38.0: Count 10
Value 52.0: Count 3
Value 65.0: Count 2
Value 26.0: Count 20
Value 32.0: Count 15
Value 63.0: Count 4
Value 56.0: Count 1
Value 46.0: Count 5
Value 20.0: Count 6
Value 22.0: Count 7
Value 40.0: Count 6
Value 61.0: Count 1
Value 19.0: Count 1
Value 54.0: Count 2
Value 60.0: Count 2
Value 74.0: Count 1
Value 50.0: Count 3
Value 64.0: Count 1
Value 59.0: Count 1
Value 55.0: Count 1
Value 43.0: Count 5
Value 57.0: Count 2

Common values for Age in years in Negative Class:
Value 22.0: Count 11
Value 53.0: Count 5
Value 28.0: Count 15
Value 25.0: Count 18
Value 24.0: Count 19
Value 60.0: Count 2
Value 32.0: Count 9
Value 44.0: Count 5
Value 63.0: Count 1
Value 37.0: Count 8
Value 58.0: Count 2
Value 57.0: Count 2
Value 52.0: Count 1
Value 23.0: Count 19
Value 61.0: Count 3
Value 40.0: Count 6
Value 34.0: Count 11
Value 47.0: Count 5
Value 33.0: Count 13
Value 39.0: Count 6
Value 30.0: Count 11
Value 27.0: Count 12
Value 29.0: Count 15
Value 66.0: Count 2
Value 20.0: Count 5
Value 31.0: Count 11
Value 26.0: Count 14
Value 36.0: Count 6
Value 74.0: Count 1
Value 54.0: Count 2
Value 21.0: Count 5
Value 50.0: Count 3
Value 45.0: Count 3
Value 48.0: Count 3
Value 42.0: Count 8
Value 38.0: Count 4
Value 46.0: Count 4
Value 35.0: Count 6
Value 55.0: Count 3
Value 41.0: Count 4
Value 43.0: Count 5
Value 65.0: Count 1
Value 19.0: Count 1
Value 59.0: Count 1
Value 51.0: Count 1
Value 68.0: Count 1
Value 49.0: Count 1

Common values for Other installment plans in Positive Class:
Value 2.0: Count 260
Value 0.0: Count 39
Value 1.0: Count 13

Common values for Other installment plans in Negative Class:
Value 2.0: Count 220
Value 1.0: Count 19
Value 0.0: Count 55

Common values for Housing in Positive Class:
Value 1.0: Count 238
Value 2.0: Count 27
Value 0.0: Count 47

Common values for Housing in Negative Class:
Value 1.0: Count 182
Value 2.0: Count 42
Value 0.0: Count 70

Common values for Number of existing credits at this bank in Positive Class:
Value 1.0: Count 193
Value 3.0: Count 4
Value 2.0: Count 112
Value 4.0: Count 3

Common values for Number of existing credits at this bank in Negative Class:
Value 1.0: Count 195
Value 2.0: Count 91
Value 3.0: Count 6
Value 4.0: Count 2

Common values for Job in Positive Class:
Value 1.0: Count 57
Value 2.0: Count 205
Value 3.0: Count 44
Value 0.0: Count 6

Common values for Job in Negative Class:
Value 2.0: Count 185
Value 3.0: Count 47
Value 1.0: Count 56
Value 0.0: Count 6

Common values for Number of people being liable to provide maintenance for in Positive Clas
Value 2.0: Count 39
Value 1.0: Count 273

Common values for Number of people being liable to provide maintenance for in Negative Clas
Value 1.0: Count 248
Value 2.0: Count 46

Common values for Telephone in Positive Class:
Value 0.0: Count 176
Value 1.0: Count 136

Common values for Telephone in Negative Class:
Value 0.0: Count 187
Value 1.0: Count 107

Common values for Foreign worker in Positive Class:
Value 0.0: Count 304
Value 1.0: Count 8

Common values for Foreign worker in Negative Class:
Value 0.0: Count 290
Value 1.0: Count 4

[Done] exited with code=0 in 2.436 seconds
```