Binary Classification

For different datasets

Chiorean Bogdan-Alin

Table of contents

Table of contents

01

Problem statement

Problem statement

- Two different datasets with particular circumstances and difficulties

- The first dataset covers a wide variety of features related to issuing insurances
- The second dataset, which has fewer attributes, relates to credit card approvals.

For each dataset, we will perform binary classification using various algorithms. The performance of these algorithms will be rigorously evaluated using established measurement metrics to determine their effectiveness and suitability for the respective tasks. This approach ensures a comprehensive understanding of predictive accuracy across varied scenarios.

02

Datasets description

Dataset description

1. Insurance Dataset (CoIL Challenge 2000)

- The first dataset is a small-sized collection of insurance data, focusing on applicants' information. Each record comprises 86 attributes, divided into the categories: **Sociodemographic data (Attributes 1–43),** derived from zip codes, meaning all individuals within the same zip code share identical sociodemographic features, and **Product ownership data (Attributes 44–86),** indicating ownership of various insurance products.
- The target variable is **Attribute 86 ("CARAVAN: Number of mobile home policies")**, representing whether an individual has mobile home policies.

- Number of columns: **86**
- Number of records: **5822 (train + validation)**
- Particularities:
    - Data is split into separate files (train + val, test)
    - Labeled data

Dataset description

2. Credit Card Approval Prediction (Kaggle)

- This dataset involves credit card applicant data used to predict whether an applicant is a 'good' or 'bad' client. Credit scoring models, traditionally built using logistic regression for transparency and binary classification, have been enhanced by machine learning methods like Boosting, Random Forest, and Support Vector Machines. However, these advanced methods often lack interpretability for customers and regulators. The task requires constructing a machine learning model to classify applicants, addressing challenges like undefined 'good' or 'bad' labels (requiring techniques like vintage analysis) and handling significant class imbalance in the data.

- Number of columns: **18 + 3**
- Number of records: **438510**
- Particularities:
    - Dataset split into 2 .csv files
    - Unlabeled data

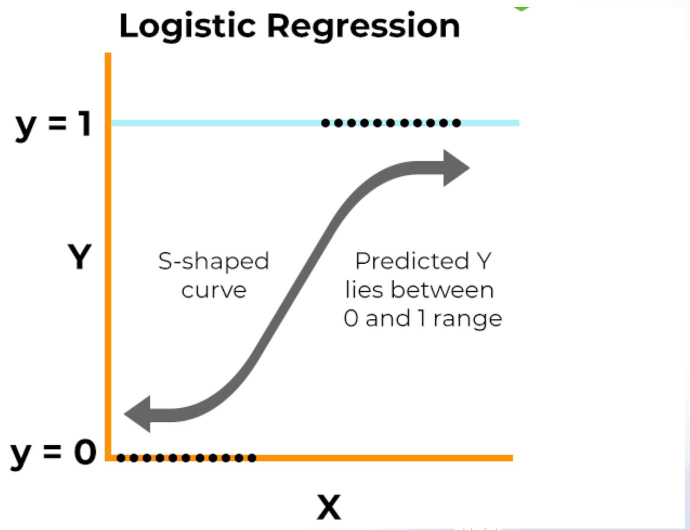
03

Theoretical foundations

Theoretical foundations

1. Logistic regression

- Logistic Regression is a statistical and machine learning algorithm used for **binary classification** problems. It predicts the probability that an input belongs to a specific class by applying the **logistic (sigmoid) function** to a linear combination of input features. The output probability is thresholded (e.g., >0.5) to determine the predicted class.



Theoretical foundations

1. Logistic regression – WHY?

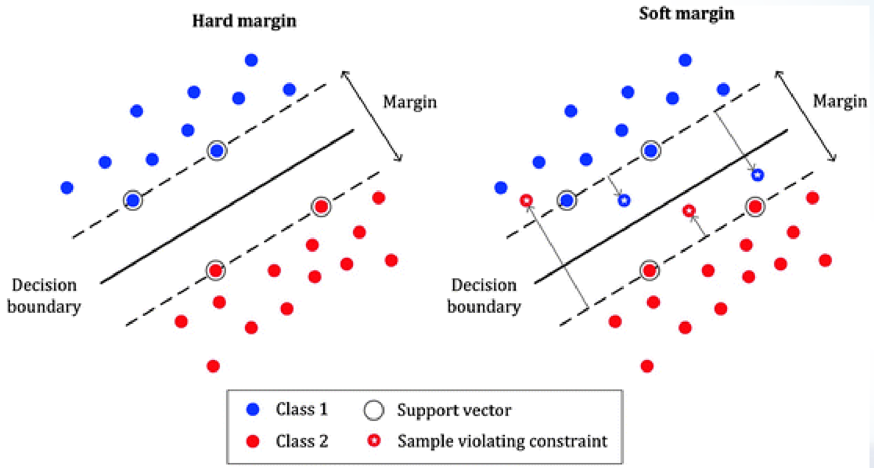
- Logistic regression is widely used due to its:
    - simplicity,
    - efficiency,
    - and interpretability,

  as it provides clear coefficients that quantify the impact of each feature on the outcome.

Theoretical foundations

2. Soft-Margin SVM (Support Vector Machine)

- A Soft-Margin Support Vector Machine (SVM) is a supervised machine learning algorithm used for binary classification. The algorithm introduces a relaxation to handle situations where some misclassification is permitted, in contrast to Hard-Margin SVM, which needs fully linearly separable data. This is accomplished by adding slack variables ξi, which permit certain points to fall inside the margin or be incorrectly classified. Using a regularization parameter C, the objective is still to identify the best hyperplane that minimizes misclassification errors and maximizes the margin.



Theoretical foundations

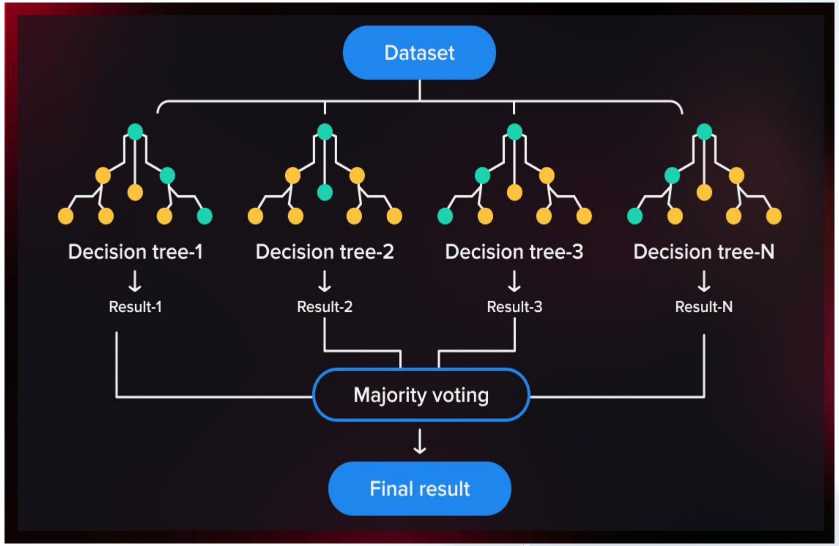
2. Soft-Margin SVM (Support Vector Machine) – WHY?

- Choosing Soft-Margin SVM for this project came more as a curiosity, because this technique requires that data is linearly, and strictly separable in the feature space.
- We also want to see if there are many outliers in our provided datasets, as SVM is highly-sensible to them

Theoretical foundations

3. Random Forest

- Random Forest is an ensemble learning method used for classification and regression tasks. It builds multiple decision trees during training and combines their outputs to improve predictive accuracy and reduce overfitting. Each tree is trained on a random subset of the data and features, introducing diversity that enhances the model's robustness. For classification, the final prediction is determined by a majority vote across all trees, while for regression, it averages the outputs.



Theoretical foundations

3. Random Forest – WHY?

- Random Forest is valued for its:
  - flexibility,
  - robustness to noise and outliers,
  - high performance on complex datasets,
  - ability to handle missing data,
  - scalability
  - and feature importance estimation.

04

Design& implementation

Design & implementation

General aspects

- **Chosen environment**: Google Colab
- **Implementations**: from ML laboratory
- **Evaluation & Metrics**: Python libraries

Why?

- **Google Colab**: cloud-based environment, accessibility
- **Laboratory-implemented algorithms**: hands-on development, ease of following instructions
- **Python libraries**: extensive library ecosystem, ease of learning

Design & implementation

About implementation

Small dataset

- First try: adding categorical columns into consideration
- Second try: dropping low-correlation degree columns
- Third try: original data

Big dataset

- First try: inserting categorical columns into evaluation
- Second try: combining columns, based on correlation degree
- Third try: corrected mistakes from the first dataset + combined columns, based on correlation degree

Design & implementation

Datasets particularities

Small dataset

- Only numerical values, a dictionary is provided for categorical columns
- Target value distribution: more details later
- High number of correlated columns (46 pairs with > 0.5 correlation grade)

Big dataset

- Missing values (Null) for a column
- Not fully understood dataset (two .csv files, duplicates, common ID count smaller than actual dataset)
- Unlabeled data: test were performed using pseudo-labeling

05

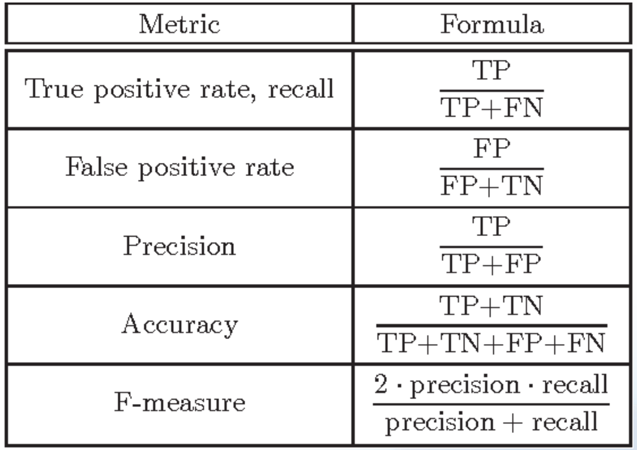
Evaluation

Evaluation

- For this project, besides the basic metrics offered in the implementation of the algorithms, such as accuracy and loss, I also included some other metrics, using the Sklearn library. The included evaluation metrics are:
    - Precision
    - Recall
    - F1-Score

  , as well as:
    - Confusion Matrix

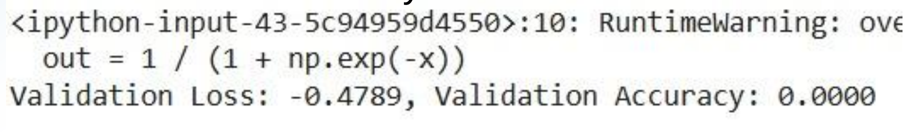
- The achieved results are presented in the next slides



Metric	Formula
True positive rate, recall	$\frac{\text{TP}}{\text{TP+FN}}$
False positive rate	$\frac{\text{FP}}{\text{FP+TN}}$
Precision	$\frac{\text{TP}}{\text{TP+FP}}$
Accuracy	$\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$
F-measure	$\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

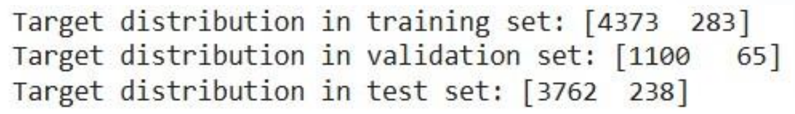
Evaluation – Small dataset

First issue: weird accuracy



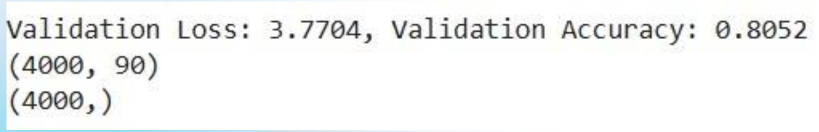
```
<ipython-input-43-5c94959d4550>:10: RuntimeWarning: ove
  out = 1 / (1 + np.exp(-x))
Validation Loss: -0.4789, Validation Accuracy: 0.0000
```

Target particularity



```
Target distribution in training set: [4373   283]
Target distribution in validation set: [1100    65]
Target distribution in test set: [3762   238]
```

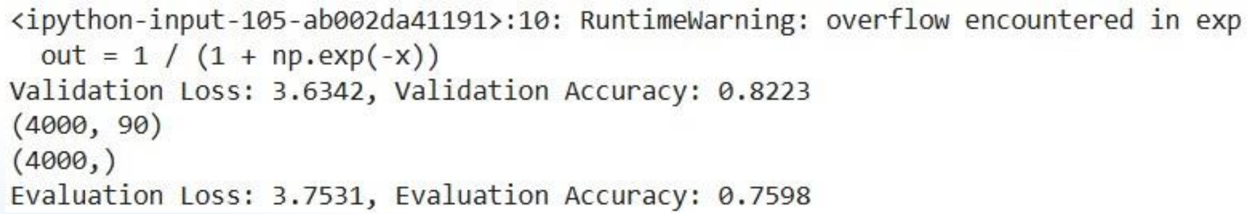
First successful run on Logistic Regression



```
Validation Loss: 3.7704, Validation Accuracy: 0.8052
(4000, 90)
(4000,)
```

Evaluation – Small dataset

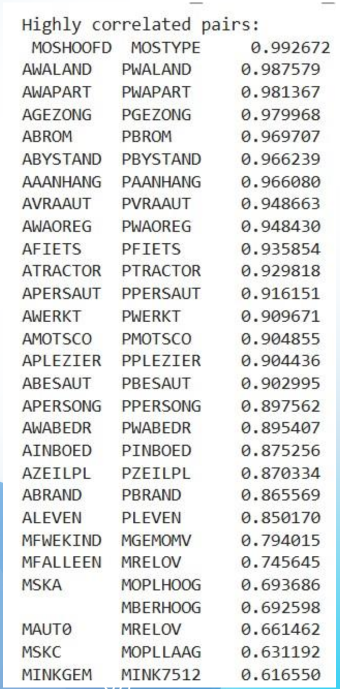
Second try – Logistic Regression



```
<ipython-input-105-ab002da41191>:10: RuntimeWarning: overflow encountered in exp
  out = 1 / (1 + np.exp(-x))
Validation Loss: 3.6342, Validation Accuracy: 0.8223
(4000, 90)
(4000,)
Evaluation Loss: 3.7531, Evaluation Accuracy: 0.7598
```

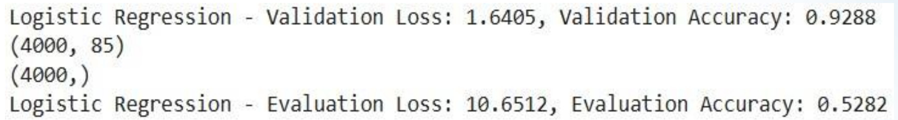
Evaluation – Small dataset

Corellation between columns

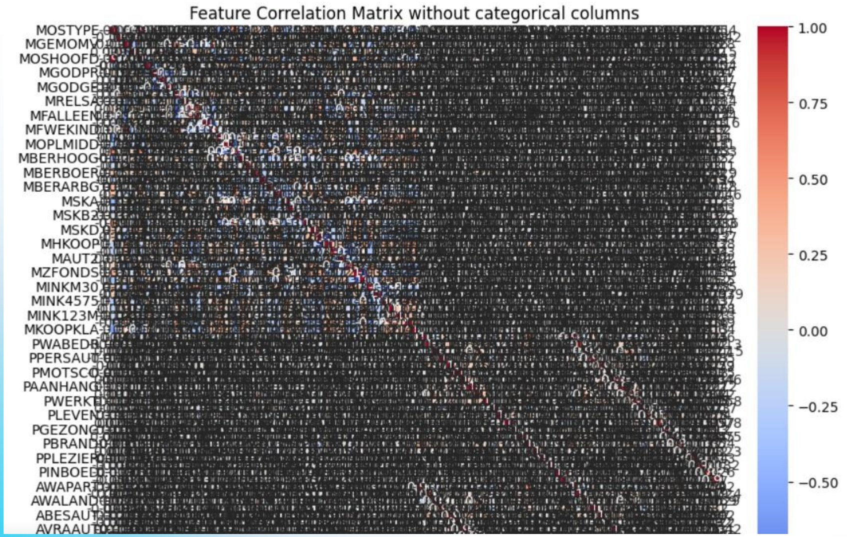


```
Highly correlated pairs:
 MOSHOOFD  MOSTYPE     0.992672
AWALAND   PWALAND     0.987579
AWAPART   PWAPART     0.981367
AGEZONG   PGEZONG     0.979968
ABROM     PBROM       0.969707
ABYSTAND  PBYSTAND    0.966239
AAANHANG  PAANHANG    0.966080
AVRAAUT   PVRAAUT     0.948663
AWAOREG   PWAOREG     0.948430
AFIETS    PFIETS      0.935854
ATRACTOR  PTRACTOR    0.929818
APERSAUT  PPERSAUT    0.916151
AWERKT    PWERKT      0.909671
AMOTSCO   PMOTSCO     0.904855
APLEZIER  PPLEZIER    0.904436
ABESAUT   PBESAUT     0.902995
APERSONG  PPERSONG    0.897562
AWABEDR   PWABEDR     0.895407
AINBOED   PINBOED     0.875256
AZEILPL   PZEILPL     0.870334
ABRAND    PBRAND      0.865569
ALEVEN    PLEVEN      0.850170
MFWEKIND  MGEMOMV     0.794015
MFALLEEN  MRELOV      0.745645
MSKA      MOPLHOOG    0.693686
          MBERHOOG    0.692598
MAUT0     MRELOV      0.661462
MSKC      MOPLLAAG    0.631192
MINKGEM   MINK7512    0.616550
```

Final try – Logistic Regression

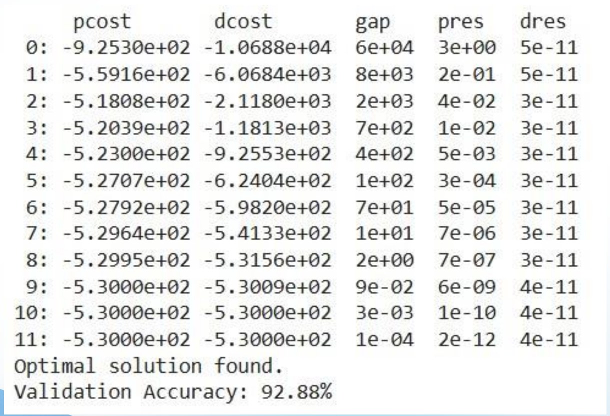


```
Logistic Regression - Validation Loss: 1.6405, Validation Accuracy: 0.9288
(4000, 85)
(4000,)
Logistic Regression - Evaluation Loss: 10.6512, Evaluation Accuracy: 0.5282
```

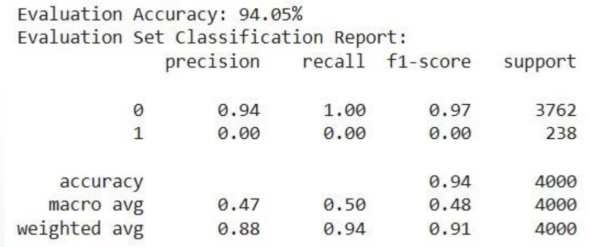


Evaluation – Small dataset

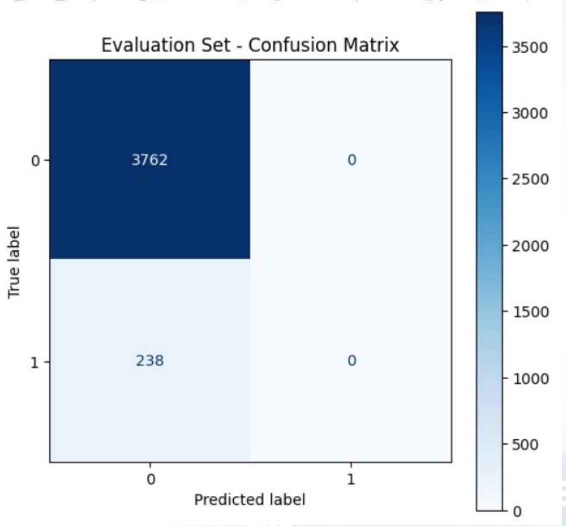
Final try – Soft-Margin SVM



```
     pcost       dcost       gap    pres   dres
 0: -9.2530e+02 -1.0688e+04  6e+04  3e+00  5e-11
 1: -5.5916e+02 -6.0684e+03  8e+03  2e-01  5e-11
 2: -5.1808e+02 -2.1180e+03  2e+03  4e-02  3e-11
 3: -5.2039e+02 -1.1813e+03  7e+02  1e-02  3e-11
 4: -5.2300e+02 -9.2553e+02  4e+02  5e-03  3e-11
 5: -5.2707e+02 -6.2404e+02  1e+02  3e-04  3e-11
 6: -5.2792e+02 -5.9820e+02  7e+01  5e-05  3e-11
 7: -5.2964e+02 -5.4133e+02  1e+01  7e-06  3e-11
 8: -5.2995e+02 -5.3156e+02  2e+00  7e-07  3e-11
 9: -5.3000e+02 -5.3009e+02  9e-02  6e-09  4e-11
10: -5.3000e+02 -5.3000e+02  3e-03  1e-10  4e-11
11: -5.3000e+02 -5.3000e+02  1e-04  2e-12  4e-11
Optimal solution found.
Validation Accuracy: 92.88%
```

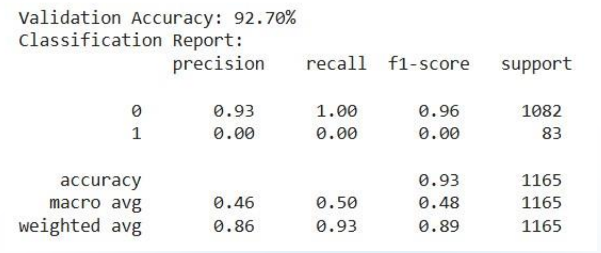


```
Evaluation Accuracy: 94.05%
Evaluation Set Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      3762
           1       0.00      0.00      0.00       238

    accuracy                           0.94      4000
   macro avg       0.47      0.50      0.48      4000
weighted avg       0.88      0.94      0.91      4000
```

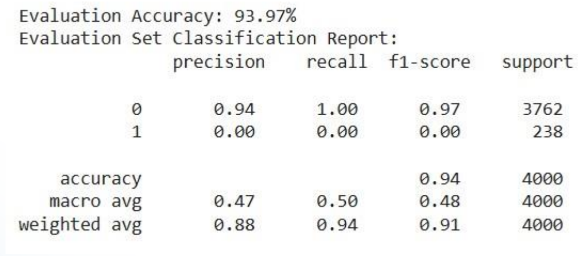


Evaluation – Small dataset

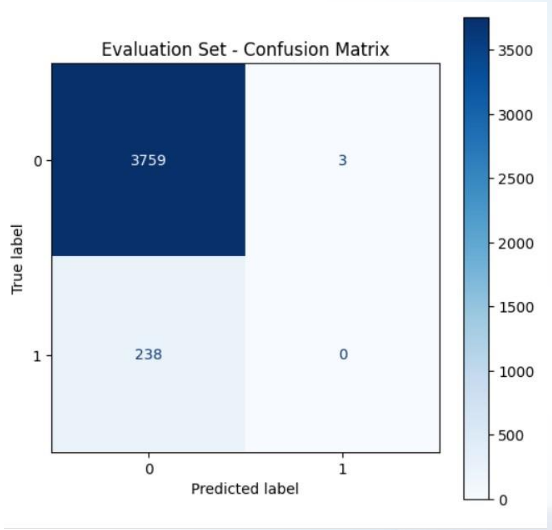
Final try – Random Forest



```
Validation Accuracy: 92.70%
Classification Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96      1082
           1       0.00      0.00      0.00        83

    accuracy                           0.93      1165
   macro avg       0.46      0.50      0.48      1165
weighted avg       0.86      0.93      0.89      1165
```

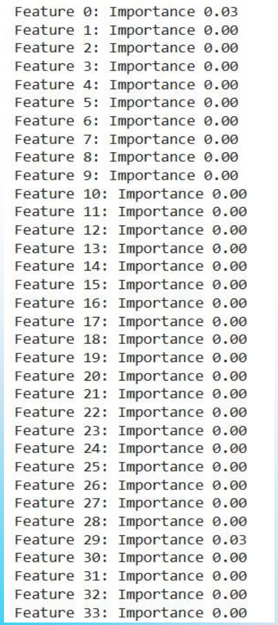


```
Evaluation Accuracy: 93.97%
Evaluation Set Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      3762
           1       0.00      0.00      0.00       238

    accuracy                           0.94      4000
   macro avg       0.47      0.50      0.48      4000
weighted avg       0.88      0.94      0.91      4000
```

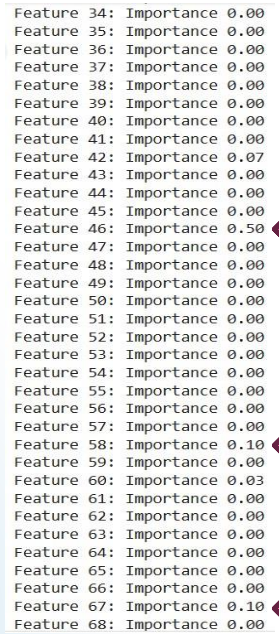


Evaluation – Small dataset

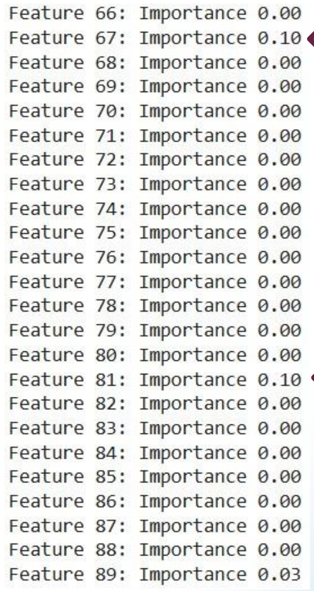
Final try – Random Forest – Feature importance



```
Feature 0: Importance 0.03
Feature 1: Importance 0.00
Feature 2: Importance 0.00
Feature 3: Importance 0.00
Feature 4: Importance 0.00
Feature 5: Importance 0.00
Feature 6: Importance 0.00
Feature 7: Importance 0.00
Feature 8: Importance 0.00
Feature 9: Importance 0.00
Feature 10: Importance 0.00
Feature 11: Importance 0.00
Feature 12: Importance 0.00
Feature 13: Importance 0.00
Feature 14: Importance 0.00
Feature 15: Importance 0.00
Feature 16: Importance 0.00
Feature 17: Importance 0.00
Feature 18: Importance 0.00
Feature 19: Importance 0.00
Feature 20: Importance 0.00
Feature 21: Importance 0.00
Feature 22: Importance 0.00
Feature 23: Importance 0.00
Feature 24: Importance 0.00
Feature 25: Importance 0.00
Feature 26: Importance 0.00
Feature 27: Importance 0.00
Feature 28: Importance 0.00
Feature 29: Importance 0.03
Feature 30: Importance 0.00
Feature 31: Importance 0.00
Feature 32: Importance 0.00
Feature 33: Importance 0.00
```



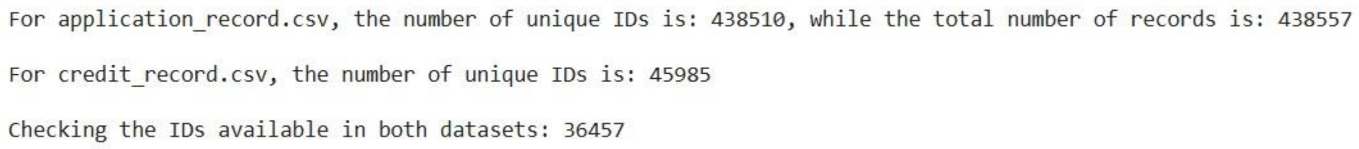
```
Feature 34: Importance 0.00
Feature 35: Importance 0.00
Feature 36: Importance 0.00
Feature 37: Importance 0.00
Feature 38: Importance 0.00
Feature 39: Importance 0.00
Feature 40: Importance 0.00
Feature 41: Importance 0.00
Feature 42: Importance 0.07
Feature 43: Importance 0.00
Feature 44: Importance 0.00
Feature 45: Importance 0.00
Feature 46: Importance 0.50
Feature 47: Importance 0.00
Feature 48: Importance 0.00
Feature 49: Importance 0.00
Feature 50: Importance 0.00
Feature 51: Importance 0.00
Feature 52: Importance 0.00
Feature 53: Importance 0.00
Feature 54: Importance 0.00
Feature 55: Importance 0.00
Feature 56: Importance 0.00
Feature 57: Importance 0.00
Feature 58: Importance 0.10
Feature 59: Importance 0.00
Feature 60: Importance 0.03
Feature 61: Importance 0.00
Feature 62: Importance 0.00
Feature 63: Importance 0.00
Feature 64: Importance 0.00
Feature 65: Importance 0.00
Feature 66: Importance 0.00
Feature 67: Importance 0.10
Feature 68: Importance 0.00
```



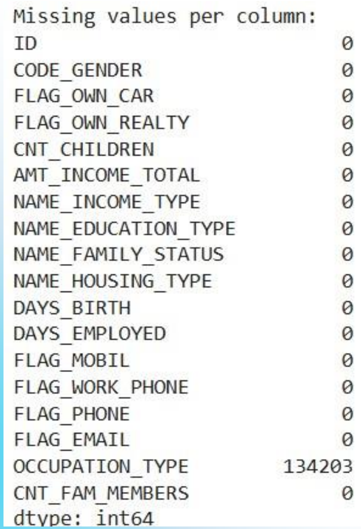
```
Feature 66: Importance 0.00
Feature 67: Importance 0.10
Feature 68: Importance 0.00
Feature 69: Importance 0.00
Feature 70: Importance 0.00
Feature 71: Importance 0.00
Feature 72: Importance 0.00
Feature 73: Importance 0.00
Feature 74: Importance 0.00
Feature 75: Importance 0.00
Feature 76: Importance 0.00
Feature 77: Importance 0.00
Feature 78: Importance 0.00
Feature 79: Importance 0.00
Feature 80: Importance 0.00
Feature 81: Importance 0.10
Feature 82: Importance 0.00
Feature 83: Importance 0.00
Feature 84: Importance 0.00
Feature 85: Importance 0.00
Feature 86: Importance 0.00
Feature 87: Importance 0.00
Feature 88: Importance 0.00
Feature 89: Importance 0.03
```

Evaluation –Big dataset

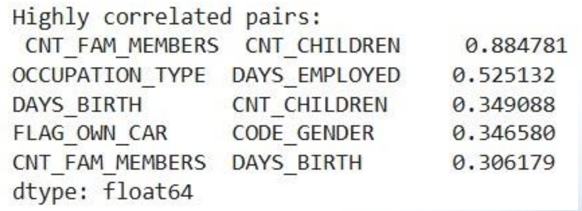
Dataset particularities



```
For application_record.csv, the number of unique IDs is: 438510, while the total number of records is: 438557

For credit_record.csv, the number of unique IDs is: 45985

Checking the IDs available in both datasets: 36457
```



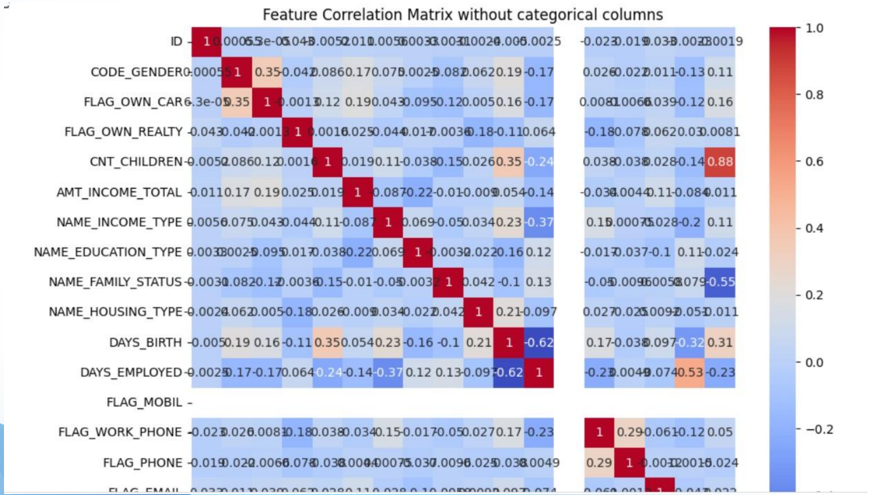
```
Missing values per column:
ID                          0
CODE_GENDER                 0
FLAG_OWN_CAR                0
FLAG_OWN_REALTY             0
CNT_CHILDREN                0
AMT_INCOME_TOTAL            0
NAME_INCOME_TYPE            0
NAME_EDUCATION_TYPE         0
NAME_FAMILY_STATUS          0
NAME_HOUSING_TYPE           0
DAYS_BIRTH                  0
DAYS_EMPLOYED               0
FLAG_MOBIL                  0
FLAG_WORK_PHONE             0
FLAG_PHONE                  0
FLAG_EMAIL                  0
OCCUPATION_TYPE        134203
CNT_FAM_MEMBERS             0
dtype: int64
```

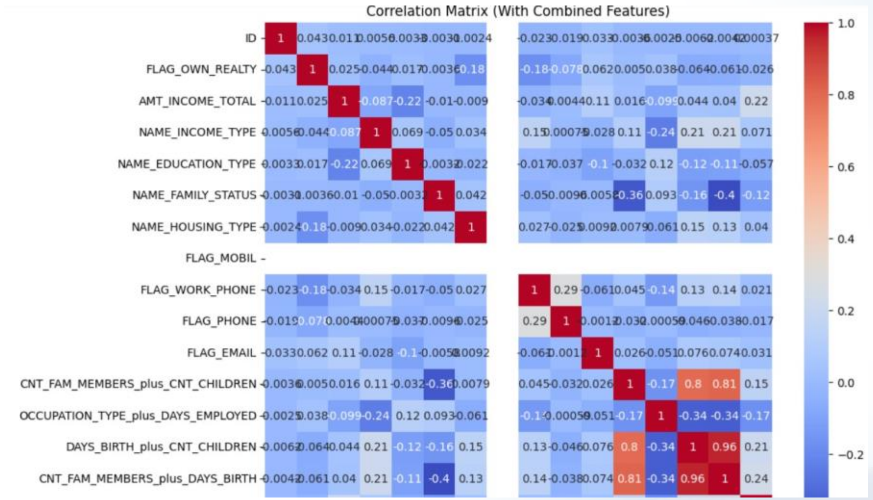


```
Highly correlated pairs:
 CNT_FAM_MEMBERS  CNT_CHILDREN     0.884781
OCCUPATION_TYPE  DAYS_EMPLOYED    0.525132
DAYS_BIRTH       CNT_CHILDREN     0.349088
FLAG_OWN_CAR     CODE_GENDER      0.346580
CNT_FAM_MEMBERS  DAYS_BIRTH       0.306179
dtype: float64
```

Evaluation –Big dataset

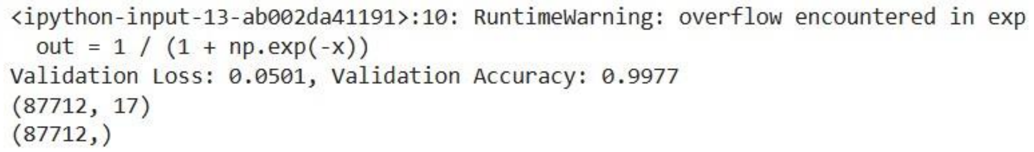
Correlation matrices





Evaluation –Big dataset

Logistic Regression



```
<ipython-input-13-ab002da41191>:10: RuntimeWarning: overflow encountered in exp
  out = 1 / (1 + np.exp(-x))
Validation Loss: 0.0501, Validation Accuracy: 0.9977
(87712, 17)
(87712,)
```

Soft-Margin SVM – broke down due to exceeding available RAM

Random Forest – still runs…

06

Analysis& interpretation

Analysis & interpretation

As seen in the previous photos, we extract the following for the **small dataset:**

- Logistic Regression:

- slow start, with high loss, but good accuracy
- for the last try, smaller loss, but still high, with a greater accuracy on evaluation, **BUT**
- huge loss, and worse accuracy on test dataset

- Soft-Margin SVM:

- Great accuracy, **BUT**
- Biased answer – determined by the target label’s distribution

- Random Forest

- Great accuracy, **BUT**
- Biased answer – determined by the target label’s distribution

Analysis & interpretation

As for the **big dataset**, we see the following:

- Logistic Regression:
• Very small loss, great accuracy

- Soft-Margin SVM:
• Broke at runtime, because of the kernel's quadratic memory requirement

- Random Forest
• Still running, even after 4 hours

07

Conclusions

Conclusions

After elaborating about the used algorithms, and their evaluation respectively, we can summarize the discussion as follows:

<u>**Small dataset**</u>

- for **logistic regression**, validation works well, but evaluation (test) fails hard, possibly indicating **overfitting**
- the idea can be backed by the target label's distribution, which is shown as imbalanced
- for **Soft-Margin SVM**, we obtained good accuracy on both validation and evaluation (test) datasets, but the answer seems biased (as shown in the confusion matrix)
- for **Random Forest**, the same conclusions as in the Soft-Margin SVM case

<u>**Big dataset**</u>

- for **logistic regression**, validation works brilliantly, but no print for evaluation (test) dataset
- for **Soft-Margin SVM**, we failed to execute the algorithm due to the kernel's quadratic memory requirement, bad for a big dataset
- for **Random Forest**, no conclusion can be made as of yet (still running)
- one important conclusion for the big dataset: whole testing was done using pseudo-labelling

08

Future work

Future work

Regarding the possibilities of further improvement, there are plenty:

- **Fine-tuning parameters**. The shown results were based on a rigid setup, mainly the one used in the laboratory work. For each algorithm, we can improve the hyperparameters using diverse techniques.
- **Data quality and challenges**. As discussed in the presentation, the particularities of each dataset directly influence the output of each algorithms. E.g. For the first dataset, data imbalance pushed the logistic regression into a possible **overfit**. For the second dataset, all results are based on a pseudo-labeled target.
- **Handle scalability**. As seen in the results, Soft-Margin SVM failed for the large dataset, due to its structure. Here, one solution can be reducing the feature dimensionality, using Principal Component Analysis(PCA)
- **Exploring advanced algorithms**. This can be done especially for the larger dataset, as traditional methods can prove slow or even unreliable

Thanks!

Do you have any questions?

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics and images by **Freepik**

Please keep this slide for attribution