Part B

INTRODUCTION

We are provided with a dataset containing 1000 entries with 20 variables where each entry represents a person who takes credit from a bank. Seven attributes are numerical and 13 categorical. Numerical attributes include the credit amount and age of the person, whereas categorical attributes include the credit history, whether that person was employed and their marital status. Each person is classified as good or bad (in two separate columns) depending on the credit risk according to the set of attributes. We deleted one column given that the features were directly opposed (a person who was classified as bad, was not classified as good). The task at hand was to split the dataset into two subsets, depending on the status of the existing checking account, select four variables which are suitable for building a scorecard, split the subsets into train and tests splits and train a linear and logistic regression model on the training set and evaluate it on the test set.

TASK 1

Before splitting the data into the subsets we inspected the data for missing values. We discovered several “X” in the purpose column and deleted those given the number was negligible and deleting those would not significantly reduce the amount of data. No further data cleaning was performed given we only selected 4 attributes for each subset, which were inspected later (see below). Sample 1 consisted of 532 entries and sample 2 of 456 entries. See the Appendix for the code behind Task 1.

TASK 2

In order to know whether a machine learning model performs well we need to test it on unseen data (data which was not seen during training). Hence, we need to split our data into multiple parts: a training set which is used to train the model, a validation set which is used to refine the model, for example, to tune hyperparameters and test the model on the validation set, and a test set which is not used for any tuning and resembles unseen, real-world data and is not touched until the very end when the model is finalised.

If we do not have a lot of data (as in our case) we want to avoid losing any data and maximise the number of instances we have for training the model. Hence, an alternative approach is to only set a test set aside but instead of also having a separate set for validation, use a process called cross-validation for hyperparameter tuning and feature selection. The process works as follows: It splits our dataset into K-folds, then the model is trained on K-1 folds and validated on the remaining one, for K iterations. So each time, because of the K rotations of the validation set, the model is trained and validated on a new composition of data. Finally, we can use the whole data to train our final model.

The issue when splitting the samples into train and test splits was the imbalance in the target variable in sample 2 (where the amount on the checking account was negative or less than 200 DM). The imbalance can be seen in Figure 1. In Sample 2 only 60 people have a bad credit risk score. This might result in the model not learning enough about this group and struggling with predicting that class.

In general, using 80% of the data as the training set, 10% as the validation set, and 10% as the test set is a common split to start with. However, due to the small size of the available data for training, I have decided to only use 10% of the data for testing and 90% for training, using cross-validation for hyperparameter tuning. The test set was stratified to ensure the same proportions of the target variable in the training and test sets.

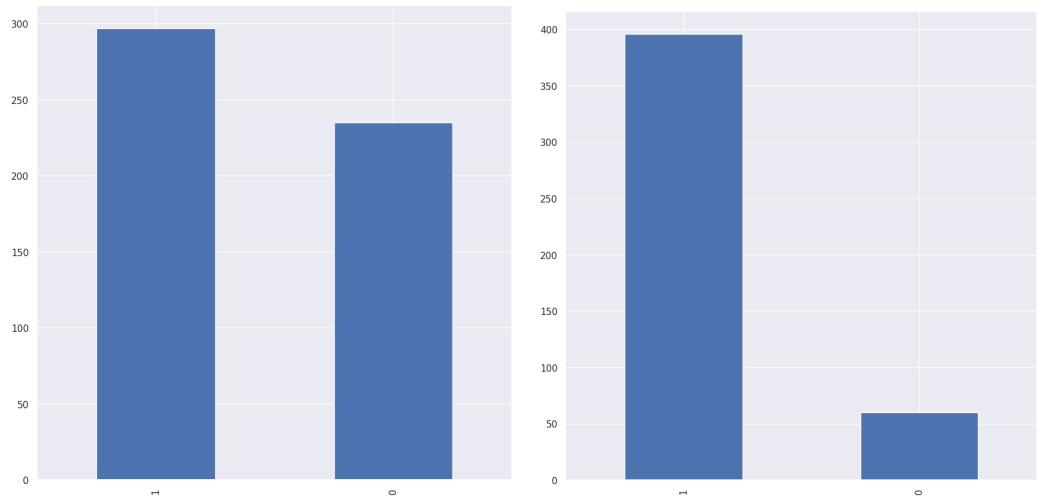

Figure 1 - Distributions of the target variable in sample 1 (left) and sample 2 (right)

TASK 3

In order to build the scorecards we performed Chi-Square tests to measure the statistically significant correlation between the attributes and the target variable. The purpose of this test is to determine if the difference between the observed and expected data is due to change or whether there is a relationship between the variables. Since we were interested in categorical values with at least three categories, we considered the following categorical attributes (strictly speaking some of these are continuous but with a very low spread, do not require binning, and hence were regarded as categorical): Checking, Job, Other, Resident, Housing, Existing Credits, Marital Status, Purpose, Employed, Savings, Coapp, Depends, History, and Property. The following three continuous variables were binned and also considered: Amount, Duration and Age. Amount was binned into five categories: 0-1000, 1000-5000, 5000-10000, 10000-15000, and 15000-20000. Age was binned into bins of 10 years, apart from the first group which was 18-30, and duration was binned into bins of 10 months. Chi-Square tests were performed between each attribute and the target variable. Attributes where the p-values which were smaller than 0.05 are shown in Tables 1 and 2. We selected the attributes Savings, History, Property and Duration to build the scorecard for Sample 1. For Sample 2 We selected Other, Purpose, Employed and History. None of the continuous variables which required binning, the p-value was below 0.05 and hence were not selected to build the scorecard.

The attributes were encoded using the one-hot encoding (dummy variables) and for each attribute one column was removed to avoid multicollinearity. The correlation between the dummy variables was inspected and the dummy variable that showed the highest correlation with another one was removed. Figure 2 shows the correlation between the Savings dummy

variables - Savings_1 and Savings_5 showed the highest correlation, so one of them was removed.

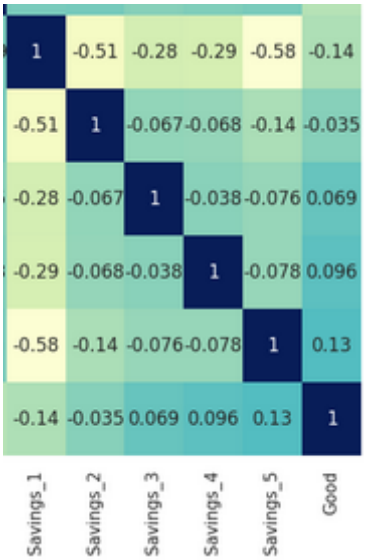

Figure 2 - Correlation between the Savings dummy variables for Sample 1

We inspected the chosen attributes for outliers (extreme values) and could not detect any. Hence, no further data cleaning was performed.

Attribute	**p-value**
Checking	0.013
Housing	0.007
Purpose	0.043
Savings	0.0008
Coapp	0.004
History	0.00003
Property	0.000009
Amount (binned)	0.019
Duration (binned)	0.000002

Table 1 - Attributes where the p-values were below 0.05 for Sample 1

Attribute	p-value
Checking	0.036
Other	0.00002
Purpose	0.008
Employed	0.004
History	0.012

Table 2 - Attributes where the p-values were below 0.05 for Sample 2

TASK 4

We fit a logistic regression to Sample 1 using the training set and set a baseline using the test set. The weighted F1 score was 0.72, and the accuracy was 72.22%. Then we performed a grid search using 10-fold cross-validation but did not find a better combination of hyperparameters than the default ones used by scikit-learn. We tried different solvers, different regularisation types and its strength. The solver used was L-BFGS with l2 regularisation and an inverse regularisation strength of 1.0.

We then fit a linear regression on the training set. The results were slightly better with a weighted F1 score of 0.73 and an accuracy of 74.07%. The confusion matrices for both models can be seen in Figure 3.

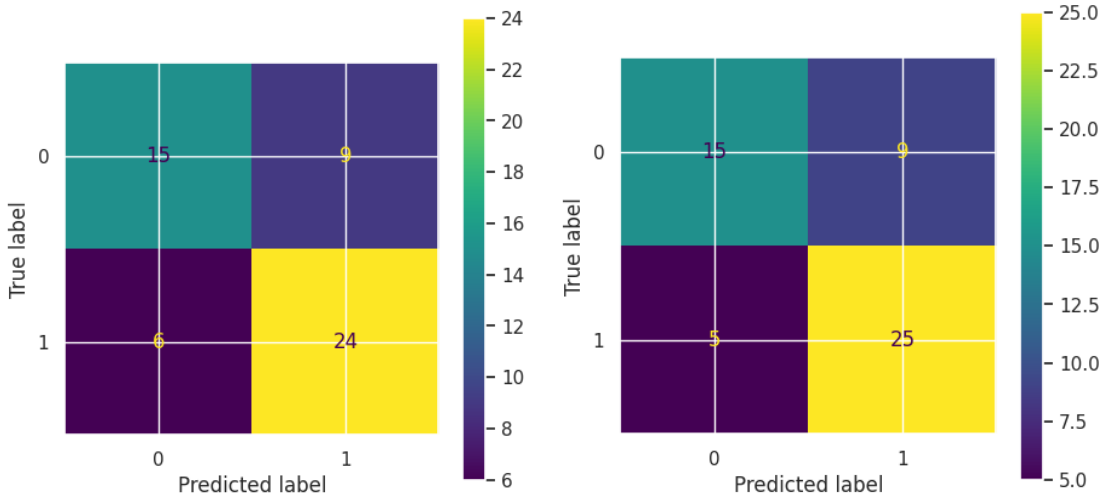

Figure 3 - Confusion Matrix for Sample 1 using Logistic Regression (left) and Linear Regression (right)

For sample 2, due to the high class imbalance, the logistic regression classifier did not manage to predict the minority class when being trained on the training set which resulted in an F1 score of 0 for the minority class and macro average F1 score of 0.59. We, therefore, downsampled the majority class. In the test set, we left the class imbalance, given it represents the real world. The test set contained 46 observations with 40 labelled as Good and 6 as Bad. Unfortunately, we did not manage to increase the performance of the model on the test set. The performance of the majority class dropped significantly and the macro

averaged F1 score dropped to 0.51. In Figure 4 one can see the confusion matrices of the test set before downsampling the training set (left) and after downsampling (left). The same issue remained when training the linear regression model which resulted in the same confusion matrix as in Figure 4 (left).

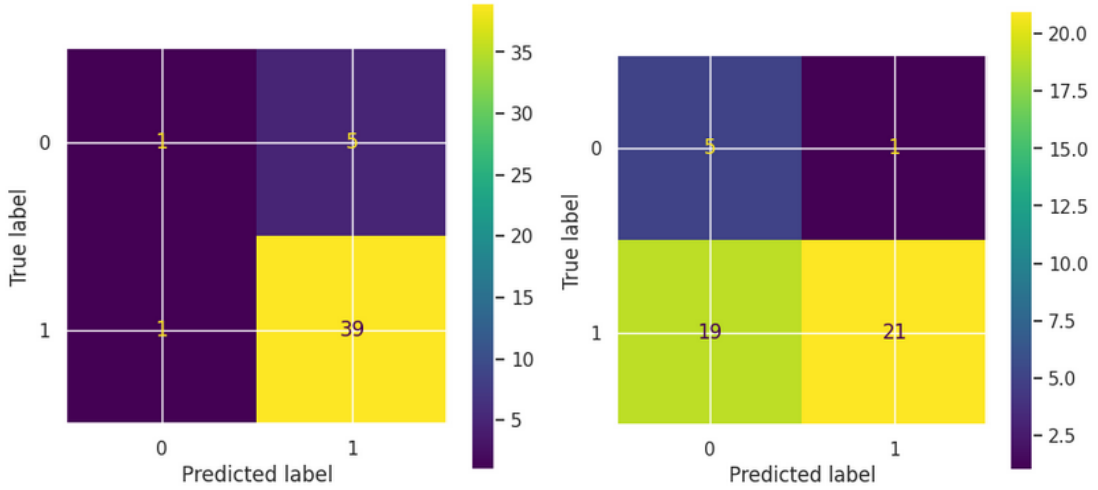

Figure 4 - Confusion Matrix for Sample 2 using Logistic Regression before downsampling (left) and after downsampling (right)

TASK 5

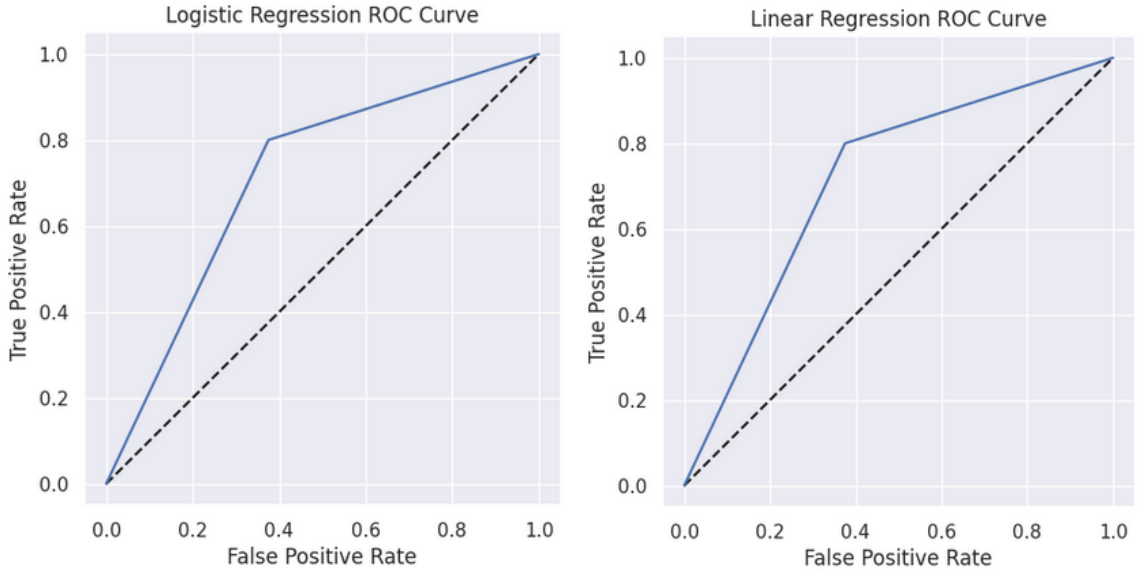

Figure 5 - ROC Curves for Sample 1 using logistic and linear regression models

As we can see in Figure 5 the ROC curves are almost identical showing no significant difference in model performance between the logistic and linear regression models. We can see that the performance of the model is not ideal but certainly not random. Unlike Sample 2, where we get very different results due to the high class imbalance. In Figure 6 we can see the ROC curves for the second sample.

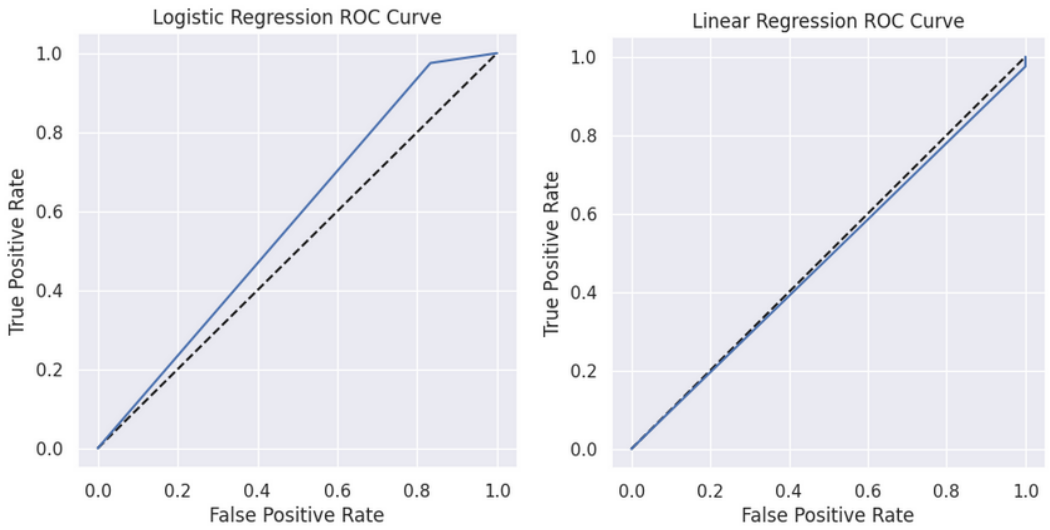

Figure 6 - ROC Curves for Sample 2 using logistic and linear regression models

The metrics for both samples can be found in Tables 3 and 4. We can see that the results for Sample 1 are much better than Sample 2. The reason for the high precision is the class imbalance where the majority of the observations from the majority class were correctly labelled. But the low recall and area under the ROC curve show that the model is useless.

Model	**Recall**	**Precision**	**ROC**	**Gini**	**KS**
Logistic	0.71	0.73	0.71	0.42	0.44
Linear	0.75	0.74	0.73	0.46	0.49

Table 3 - Metrics for Sample 1

Model	**Recall**	**Precision**	**ROC**	**Gini**	**KS**
Logistic	0.5	0.89	0.57	0.14	0.39
Linear	0.0	0.87	0.49	-0.06	-0.13

Table 4 - Metrics for Sample 2

REFERENCE LIST

Anderson, R. (2007). The Credit Scoring Toolkit. Oxford University Press.

Bellotti, A., and Crook, J. (2009). Credit scoring with macroeconomic variables using survival analysis. Journal of the Operational Research Society, 60(12), 1699-1707.

Dirick, L., Claeskens, G., and Baesens, B. (2017). Time to default in credit scoring using survival analysis: a benchmark study. Journal of the Operational Research Society, 68, 652-665.

Harrell, Jr, F. E., and Harrell, F. E. (2015). Cox proportional hazards regression model. Regression modeling strategies: With applications to linear models, logistic and ordinal regression, and survival analysis, 475-519.

Investopedia. (2020). Risk-Based Pricing. Retrieved April 30, 2023, from https://www.investopedia.com/terms/r/riskbased-pricing.asp

Lane, W. R., Looney, S. W., and Wansley, J. W. (1986). An application of the Cox proportional hazards model to bank failure. Journal of Banking & Finance, 10(4), 511-531.

Levy, J. J., and O'Malley, A. J. (2020). Don't dismiss logistic regression: the case for sensible extraction of interactions in the era of machine learning. BMC medical research methodology, 20(1), 1-15.

ListenData. (2019). Credit Risk Modelling [Online]. Available at: https://www.listendata.com/2019/08/credit-risk-modelling.html#id-fed394 [Accessed 29 April 2023].

Mukid, M. A., Azad, M. A. K., Hasan, M. T., Hasan, M. R., & Haque, M. E. (2018). Credit scoring analysis using weighted k nearest neighbor. Journal of Physics: Conference Series, 1025(1), 012114.

Schuermann, T. (2008). Credit Migration Matrices. Wiley

Siddiqi, N. (2006) Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring. Hoboken, N.J.: Wiley.

Stepanova, M., and Thomas, L. C. (2001). PHAB scores: proportional hazards analysis behavioural scores. Journal of the Operational Research Society, 52(9), 1007-1016. Palgrave Macmillan.

Thomas, L.C. (2009). Consumer Credit Models: Pricing, Profit and Portfolios. Oxford University Press.

APPENDIX

TASK 1

Checking for missing data

There are no NAN values, however, the Purpose column contains "X" - we will delete them because there are not that many instances and we would not lose a lot of data

```
[7]  1 plt.figure(figsize=(20,6))
     2 sns.displot(
     3     data=df.isna().melt(value_name="missing"),
     4     y="variable",
     5     hue="missing",
     6     multiple="fill",
     7     aspect=1
     8 )
```

```
[9]  1 df = df[df.Purpose != "X"]
```

1. Splitting the dataset

```
[10]  1 sample1 = df[(df.Checking == 1) | (df.Checking == 2)]
```

```
      1 sample2 = df[(df.Checking == 3) | (df.Checking == 4)]
```

```
[12]  1 len(sample2)

      456
```

```
[13]  1 sample1.describe().T
```

COEFFICIENTS

Logistic Regression Sample 1

	feature	value	abs_value	label
14	Savings_1	-0.917984	0.917984	Bad
13	Property_4	-0.879915	0.879915	Bad
7	duration_binned_5	-0.812115	0.812115	Bad
15	Savings_2	-0.770549	0.770549	Bad
4	duration_binned_1	0.693257	0.693257	Good
3	History_4	0.686045	0.686045	Good
6	duration_binned_4	-0.616710	0.616710	Bad
1	History_1	-0.522261	0.522261	Bad
8	duration_binned_6	-0.500801	0.500801	Bad
11	Property_2	-0.454209	0.454209	Bad
2	History_3	0.444154	0.444154	Good
0	History_0	-0.433891	0.433891	Bad
12	Property_3	-0.429398	0.429398	Bad
10	duration_binned_8	-0.421742	0.421742	Bad
17	Savings_4	0.155236	0.155236	Good
5	duration_binned_3	-0.110987	0.110987	Bad
16	Savings_3	0.007257	0.007257	Good
9	duration_binned_7	0.000000	0.000000	Bad

Logistic Regression Sample 2

	feature	value	abs_value	label
10	Emploed_1	-1.698675	1.698675	Bad
6	Purpose_5	-1.597717	1.597717	Bad
0	Other_1	-1.552770	1.552770	Bad
17	History_3	-1.221352	1.221352	Bad
3	Purpose_1	1.209695	1.209695	Good
5	Purpose_4	1.201106	1.201106	Good
14	History_0	-1.138674	1.138674	Bad
11	Emploed_2	-1.113625	1.113625	Bad
8	Purpose_8	1.047638	1.047638	Good
9	Purpose_9	-0.956034	0.956034	Bad
7	Purpose_6	-0.860804	0.860804	Bad
1	Other_2	-0.813498	0.813498	Bad
12	Emploed_3	-0.724104	0.724104	Bad
2	Purpose_0	-0.675449	0.675449	Bad
15	History_1	-0.663013	0.663013	Bad
16	History_2	-0.584244	0.584244	Bad
13	Emploed_4	0.275228	0.275228	Good
4	Purpose_2	-0.236311	0.236311	Bad

Linear Regression Sample 1

```
: lr_model.coef_
```

```
: array([-1.08245760e-01, -1.28291959e-01,  1.18467369e-01,  1.51358877e-01,
         1.36480114e-01, -3.54633348e-02, -1.68361823e-01, -2.08571946e-01,
        -2.15265830e-01,  1.66533454e-16, -5.29516710e-01, -1.12614820e-01,
        -1.02162460e-01, -2.03907471e-01, -2.23182249e-01, -1.94124487e-01,
        -2.95772160e-02,  4.08168056e-03])
```

```
: lr_model.intercept_
```

```
: 0.8258036567208189
```

Linear Regression Sample 2

```
4]: lr_model_2.coef_
```

```
4]: array([-0.21845017, -0.10745147, -0.06444493,  0.06975504, -0.0140184 ,
         0.09611964, -0.20751722, -0.08819372,  0.13396648, -0.13306085,
        -0.21234066, -0.12347348, -0.06491421,  0.01999202, -0.17239906,
        -0.02799768, -0.04275918, -0.13306828])
```

```
5]: lr_model_2.intercept_
```

```
5]: 1.004967012869712
```