Letter

Description automatically generated

4.9: Applied Machine Learning Project

A Comparison of Machine Learning Techniques for an Effective Credit Scoring Model

**Full name:** Lou Pemberton-Roberts  
**Student ID:** S21004801

**Module title:** Machine Learning (CONL708 Occurrence 1)

**Date:** 31st October 2022 (Extension: 7th November 2022)

I declare that this report is an original report of my research, has been written by me and has not been submitted for any previous degree.

## Table of Contents

1. **Introduction**
   1. The Project and Dataset
   2. The Programming Environment
2. **Data Pre-processing and Feature Extraction**
3. **Selection of Machine Learning Techniques**
   1. Support Vector Machines (SVC) Model
   2. Logistic Regression (LR) Model
   3. Decision Trees (DT) Model
4. **Model Performance Evaluation**
   1. Metrics Used for Evaluation
      1. Classification Report
      2. Confusion Matrix (Error Matrix)
      3. ROC Curve and AUC
5. **Evaluating the Models with Cross Validation (CV)**
   1. Comparison of the Three Models
   2. Classification Report for the Three Models
   3. Cross Validation (CV) Report of the Three Models
   4. Analysis of the Confusion Matrices for the Three Models
   5. Analysis of the ROC Curves for the Three Models
   6. Fine-Tuning of the Hyperparameters using GridSearchCV
6. **Conclusion with Recommendations**

**References**

# Introduction

With increased demand for credit facilities from consumers [1], comes increased risk to lenders with recent forecasts for the loan-loss ratio to rise 1.7% from an average of 1.3% in 2022 [1].

To mitigate these risks, lenders use machine learning (ML) techniques that can help to predict if a consumer is likely to pay the loan back (i.e., a good customer), or not (i.e., a bad customer), based on comparing their status to similar consumers found in historical data to output a credit score [2]. Credit scoring is a **binary classification problem** because the output will be one of two classes (i.e., ‘good’ or ‘bad’) [3].

**1.1 The Project and Dataset**

This project uses the publicly available *‘German Credit Risk – With Target’* dataset by Leonardo Ferreira [4] to build and train three machine learning classification models that use supervised learning to group consumers into binary categories; 0 (low risk customers who will repay on time) and 1 (high risk customers who may struggle to repay on time), to make a prediction for lending on a case-by-case basis.

This dataset was chosen because it contains useful information about credit data for 1000 customers, is publicly available for use, and is stored in the structured comma-separated values (CSV) format making it much quicker to pre-process and extract features from.

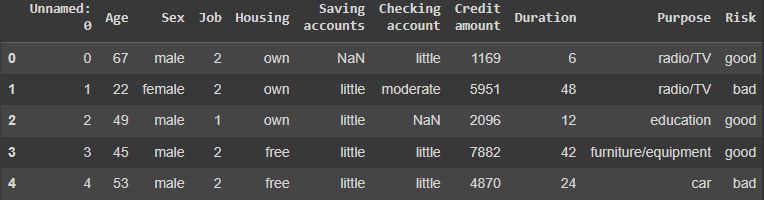
**1.2 The Programming Environment**

The programming environment chosen for the project is *Jupyter Notebooks* [5] in *Google Collaboratory Pro* [6] with *Python 3.7.15* [7]. and the final notebook is available to be viewed [8]*.*

This environment was chosen because it uses a *Linux* operating system (OS) [9] based in the cloud and provides access to GPU, more RAM, and more Disk space [6]. Python was chosen because it is a familiar programming language often used for data projects.

# Data Pre-processing and Feature Extraction

Prior to modelling, it is important to learn what the dataset is describing. The dataset is read into a *Pandas DataFrame* [10]:



**Figure 1:** German Credit Dataset [4] in Pandas DataFrame [8].

To begin pre-processing, the *‘Unnamed:0’* column is dropped since this is just a duplicate of the index itself:

A screenshot of a computer

Description automatically generated with medium confidence

**Figure 2:** German Credit Dataset [4] in Pandas DataFrame after dropping *‘Unnamed:0’* column [8].

There are some empty values included, so a check on where these values are is made and replaced with the value ‘none’:

Text

Description automatically generated

**Figure 3:** Number of NaN values in German Credit Dataset [4] [8].

A check on the unique values is made in preparation for **feature extraction**. This includes the new ‘none’ values that replaced the empty values:

Text

Description automatically generated

**Figure 4:** Unique values for each of the predictors [8].

It is now possible to identify what the predictors, target value and categories will be:

* **Predictors**: age, sex, job, housing, saving accounts, checking account, credit amount, duration, purpose.
* **Target Value**: Risk (good)
* **Categories**: ‘good’ and ‘bad’

Because it is difficult for machine learning models to process raw data [11], feature extraction is used to transform the categories into integers without losing any information from the data. For this task, **Label Encoding** [12] is used because there are only two unique categories; ‘good’ risk is mapped to 0, and ‘bad’ risk is mapped to 1:

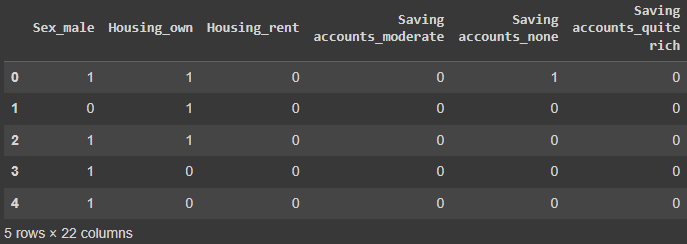


A screenshot of a computer

Description automatically generated with medium confidence

**Figure 5:** Pandas DataFrame after using label encoding [8].

One-Hot encoding [12] is then used to transform the rest of the data into dummy variables:



**Figure 6:** Section of Pandas DataFrame after using One-Hot Encoding [8].

Finally, if there is any bias or imbalance in our categories this could skew the model to a point whereby predictions will be unreliable, so a check on the distribution balance for good and bad risk is checked:

A picture containing bar chart

Description automatically generated

**Figure 7:** Balance of data distribution for categories in target [8].

Although the distribution looks unbalanced in favor of good risk (0) with a ratio of ~42% for bad risk (1), there is enough data available for the model to learn from each category, since there are 300 accounts at least for both, but the unbalance should be noted for evaluation.

The data is split 80/20 into training and testing sets ready for modelling, with the target value ‘Risk’ column dropped from the data:

Text

Description automatically generated

# Selection of Machine Learning Techniques

The machine learning techniques selected needed to be suitable for binary classification given the problem (section 1). Furthermore, the dataset is small, so the model requires low complexity [13]. With these factors in mind, it was decided that the three models used for comparison would be:

* Support Vector Classifier (SVC)
* Logistic Regression (LR)
* Decision Trees (DT)

All three models are available in Scikit-Learn [14].

* 1. **Support Vector Machines (SVC) Model**

Support Vector Classifier (SVC) is a supervised machine learning method based on *A Library for Support Vector Machines (LIBSVM)* [15]. It is used for classification problems and works by finding a hyperplane that can divide data points that have been mapped on to a high dimensional space, creating marginalized binary categories [16].

Chart, scatter chart

Description automatically generated

**Figure 8:** Support Vector Classifier (SVC) model (credit: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.)

* 1. **Logistic Regression (LR) Model**

Logistic Regression (LR) models predict a binary class based on the probability of a discrete outcome given the input variable and is also a supervised machine learning method that is available to implement through Scikit Learn libraries [14]. It is most suitable to use for binary classification problems and works by analyzing the relationship between independent variables in a dataset [17].

Chart, histogram

Description automatically generated

**Figure 9:** Logistic Regression (LR) model (credit: https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html)

* 1. **Decision Trees (DT) Model**

Decision Tree (DT) is a low-complexity, supervised machine learning model that uses entropy to calculate information gain from labelled inputs to create the best splits between leaf nodes until the data is completely divided and each of the final nodes contains just one category [19].

This makes the DT model suitable for binary classification problems and can be implemented via Scikit Learn [14].

Diagram

Description automatically generated

**Figure 10:** Decision Tree (DT) model (credit: https://scikit-learn.org/stable/modules/tree.html)

# Model Performance Evaluation

## Metrics Used for Evaluation

To evaluate the performance of each model, the following libraries were imported from Scikit Learn Metrics [14]:



* + 1. **Classification Report**

Classification Reports are used to summarize the quality of predictions made by a classification model. They use True-Positives (TP), False-Positives (FP), True-Negatives (TN), and False-Negatives (FN) to measure if a prediction was True or False, and calculate the following metrics:

* **Accuracy:** This is the ratio of correct predictions made to the sum of all predictions

Accuracy = (TP + TN) / (TP + TN + TP + FP)

Accuracy is the simplest metric and calculates how many predictions were correct.

* **Precision:** This is the ratio of true-positives to the sum of true-positives and false-positives

Precision = TP / (TP + FP)

Precision gives the proportion of positive predictions that were correct [19].

* **Recall:** This is the ratio of true-positives to the sum of true-positives and false-negatives

Recall = TP / (TP + FN)

Recall gives the proportion of actual positives that were correct and [19].

* **F1-score:** This is the harmonic mean of precision and recall, and is calculated as

F1 = 2 / (precision-1 + recall-1)

F1-score is used to measure accuracy for classification models by giving equal weight to both precision and recall metrics. This makes it more complex than the accuracy metric, and the more important metric for evaluating models with unbalanced datasets [20].

* **Support:** This is the number of samples of a class in the specified dataset [21].
  + 1. **Confusion Matrix (Error Matrix)**

A Confusion Matrix is a visualization table that summarizes the performance of binary classification models on a set of test data to quickly describe how many True-Positives (TP), False-Positives (FP), True-Negatives (TN), and False-Negatives (FN) were predicted correctly and incorrectly by the model [22].

Chart

Description automatically generated

**Figure 11:** Example of a Confusion Matrix (credit: https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826)

The goal of a good model is to have high TP and TN rates (shaded green in figure 11), with FP and FN rates (shaded red in figure 11) to be as low as possible, and uses the metrics accuracy, precision, recall and F1-score (section 1.1.1.) to evaluate performance [22].

* + 1. **ROC Curve and AUC**

The ROC Curve (receiver operating characteristic curve) and AUC (area under the receiver operating characteristics) is a graph that describes how well the model is at predicting classes.

The higher the area of AUC, the better the model [23].

The graph is plotted using three calculations:

* True Positive Rate (TPR) [23] which is the same as Recall, and calculated as

TPR = TP / (TP + FN)

* Specificity [23] which is calculated as

Specificity = TN / (TN + FP)

* And False Positive Rate (FPR) [23], which is calculated as

FPR = FP / (TN + FP)

## Evaluating the Models with Cross Validation (CV)

Cross-validation (CV) randomly splits the dataset into groups, called k-folds, and iterates through the dataset using each k-fold as the test set and the remaining sets as the training sets, until all k-folds have been the test set at least once. This allows a thorough and fair evaluation of how each model has performed but can take time to complete so is best suited to small datasets [24].



**Figure 12:** k-Fold Cross-Validation (credit: https://scikit-learn.org/stable/modules/cross\_validation.html).

To compare the models’ performance, cross\_validate was imported from the Scikit Learn Metrics model\_selection workflow library [14]:

**Text

Description automatically generated**

## Comparison of the Three Models

## Classification Report for the Three Models

The results of the Classification Report display the metrics for the SVC model, the LR model, and the DT model from top to bottom (Figure 13). Since it was established that the categories for the target value were imbalanced (section 2), the F1-score will take precedence over accuracy, as detailed in section 4.1.1.

A screenshot of a computer

Description automatically generated with low confidence

**Figure 13:** Classification Report for (top to bottom) SVC model, LR model, and DT model [8].

The SVC model has performed very poorly and is only using one classifier for predictions.

The LR model has performed good. It has the better F1-scores of all three models but has a slightly higher ratio of bad risk (1) in the support than we know there is (section 2), and the DT model is slightly lower.

## Cross Validation (CV) Report of the Three Models

The CV Report for the three models display the metrics for the SVC model, the LR model, and the DT model from top to bottom (Figure 14):

Graphical user interface, text, application

Description automatically generated

**Figure 14:** Cross-Validation for (top to bottom) SVC model, LR model, and DT model [8].

Again, there is evidence of a misclassification from the SVC model given the perfect train\_score of 1.0 but a test\_score of 0.0. However, the LR model has a low range between the train\_score and test\_score, suggesting the predictions were close to the training model.

The DT model had the best train\_score, but the test\_score was significantly lower giving too much range. However, this was also the most efficient model in time.

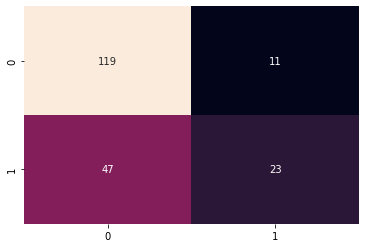
## Analysis of the Confusion Matrices for the Three Models

A picture containing graphical user interface

Description automatically generated

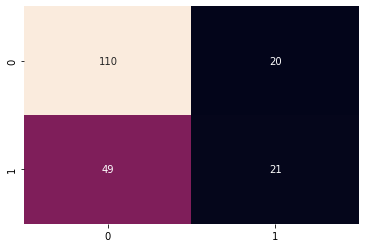
**Figure 15:** Confusion Matrix for SVC model [8].

The Confusion Matrix for the SVC model (Figure 15) predicted 130 True Negatives correctly, but 70 False Negatives. This could be due to the imbalance in the target data, but given the previous metric results, it is a suspected problem with the classifier.



**Figure 16:** Confusion Matrix for LR model [8].

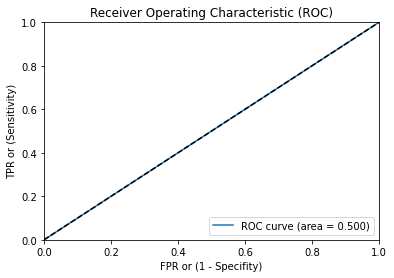
The Confusion Matrix for the LR model is better, predicting 119 True Negatives and 23 True Positives. This (~52%) is still higher than the actual rate of True Positives which was calculated as ~42% (section 2).



**Figure 17:** Confusion Matrix for DT model [8].

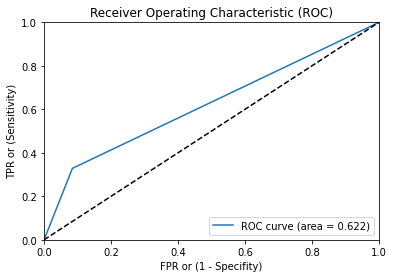
The Confusion Matrix for the DT model ranks in the middle of the three models. It also a True Positive of ~52% like the LR model, but the number of False Positives and False Negatives are higher.

## Analysis of the ROC Curves for the Three Models



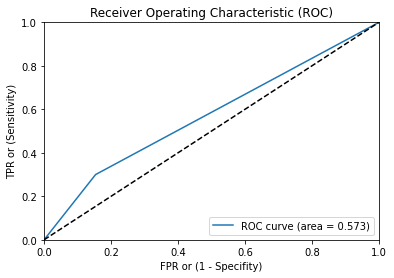
**Figure 18:** ROC curve for SVC model [8].

The ROC curve for the SVC model shows no discrimination [25] and that it cannot distinguish between the categories in the training set, therefore making it worthless.



**Figure 19:** ROC curve for LR model [8].

The ROC curve for the LR model has an acceptable level of discrimination [25] and indicates a good model.



**Figure 20:** ROC curve for DT model [8].

The ROC curve for the DT model shows poor discrimination [25], due to the large range between the TPR and FPR.

## Fine-Tuning of the Hyperparameters using GridSearchCV

To enhance the models, GridSearchCV was used from the model\_selection workflow in Scikit Learn [14] to give the best parameters for each model [26][27][28]:

Text

Description automatically generated

Grid Search [29] was used rather than Random Search [30] to ensure that all values are tested in sequence, and no values are missed from the available parameters.

# Conclusion with Recommendations

In conclusion, the Logistic Regression (LR) model was the best model for the given problem. It is felt that any shortcomings with the models were likely human error on my part, and with some further pre-processing on feature extraction being made, a revisit of the fine-tuning of the parameters, and a larger dataset the LR model could be improved further.

The SVC model was disappointing. But, again, this is likely human error in scaling prior to fine-tuning or not extracting the features correctly.

*Word count: 2,191*

## References

[1] Luttig, V. (2022). Ernst & Young Global Limited. *'Cost of living pressures to drive significant rise in UK credit card borrowing*'. EY. 4th May. Online. Available from: https://www.ey.com/en\_uk/news/2022/05/cost-of-living-pressures-to-drive-significant-rise-in-uk-credit-card-borrowing Accessed 21st October 2022.

[2] Kennedy, K. (2013). *'Credit scoring using machine learning.*' Doctoral thesis. Technological University Dublin. Online. Available from: doi:10.21427/D7NC7J Accessed 21st October 2022.

[3] Guidolin, M., Pedio, M. (2021). *‘Sharpening the Accuracy of Credit Scoring Models with Machine Learning Algorithms’*. In: Consoli, S., Reforgiato Recupero, D., Saisana, M. (eds) Data Science for Economics and Finance. Springer, Cham. Online. Available from: https://doi.org/10.1007/978-3-030-66891-4\_5 Accessed 21st October 2022.

[4] Ferreira, L. (2017) *'German Credit Risk - With Target'*. Kaggle. Online. Available from: https://www.kaggle.com/datasets/kabure/german-credit-data-with-risk Accessed 12th October 2022

[5] Kluyver, T. et al., 2016. Jupyter Notebooks – a publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt, eds. Positioning and Power in Academic Publishing: Players, Agents and Agendas. pp. 87–90.

[6] Google, Inc. (2018). *'Colaboratory: Frequently Asked Questions'*. June. Online. Available from: https://research.google.com/colaboratory/faq.html.

[7] Python Software Foundation. (2022). Python 3.7.15. October 11th. Online. Available from: https://www.python.org/downloads/release/python-3715/

[8] Pemberton-Roberts, L. (2022). *'Fine-Tuned\_Final\_Comparison\_of\_Models\_for\_Credit\_Scoring.ipynb'*. Google Colab Pro. Online. Available from: https://colab.research.google.com/drive/1SCZNuoDK7Zt3VSmN5kl1-sW-EaN0aKAV?usp=sharing

[9] Linux Kernel Organization, Inc. (2022). The Linux Kernel Archives. Online. Available from: https://www.kernel.org/

[10] NumFOCUS, Inc. (2022). *'pandas.DataFrame'*. Pandas. Online. Available from: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

[11] Weinmeister, K. (2022). *'From raw data to machine learning model, no coding required'*. Google Cloud Blog. April 14th. Online. Available from: https://cloud.google.com/blog/products/ai-machine-learning/from-raw-data-to-machine-learning-model-no-coding-required Accessed 14th October 2022.

[12] Yadav, D. (2019). *'Categorical encoding using Label-Encoding and One-Hot-Encoder'*. TDS. Medium. December 6th. Online. Available from: https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd Accessed 14th October 2022

[13] Zhang, Y., Ling, C. (2018). *'A strategy to apply machine learning to small datasets in materials science'*. NPJ Comput Mater 4, 25 (2018). Online. Available from: https://doi.org/10.1038/s41524-018-0081-z Accessed 15th October 2022.

[14] Pedregosa et al. (2011). 'Scikit-learn: Machine Learning in Python'. JMLR 12, pp. 2825-2830.

[15] Chih-Chung Chang and Chih-Jen Lin. (2011). *'LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology'*. 2:27:1--27:27. Software Available from: http://www.csie.ntu.edu.tw/~cjlin/libsvm

[16] Ian H. Witten, Eibe Frank, Mark A. Hall. (2011). *'Chapter 6 - Implementations: Real Machine Learning Schemes*'. Editor(s): Ian H. Witten, Eibe Frank, Mark A. Hall. In The Morgan Kaufmann Series in Data Management Systems. Data Mining: Practical Machine Learning Tools and Techniques (Third Edition). Morgan Kaufmann. Pages 191-304. ISBN 9780123748560. Online. Available from: https://doi.org/10.1016/B978-0-12-374856-0.00006-7. Accessed 15th October 2022.

[17] IBM (2022). *'What is logistic regression?*'. Online. Available from: https://www.ibm.com/topics/logistic-regression Accessed 15th October 2022.

[18] Ahmed, B. (2015). *'Decision Trees'.* College of Computer and Information Science. CS 6140: Machine Learning. Online. Available from: https://www.ccs.neu.edu/home/vip/teach/MLcourse/1\_intro\_DT\_RULES\_REG/lecture\_notes/lectureNotes\_DecisionTree\_Spring15.pdf Accessed 15th October 2022.

[19] Google Developers. (2022). *'Classification: Precision and Recall'*. Machine Learning. Foundational courses. Classification. Online. Available from: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall Accessed 18th October 2022.

[20] Korstanje, J. (2021). *'The F1 score'*. TDS. Medium. August 31st. Online. Available from: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6 Accessed 18th October 2022.

[21] The scikit-yb developers. (2016-1019). *'Classification Report'*. Online. Available from: https://www.scikit-yb.org/en/latest/api/classifier/classification\_report.html Accessed 18th October 2022.

[22] Ajay Kulkarni, Deri Chong, Feras A. Batarseh. (2020). *'5 - Foundations of data imbalance and solutions for a data democracy'*. Data Democracy. Academic Press. Editor(s): Feras A. Batarseh, Ruixin Yang. Pages 83-106. ISBN 9780128183663. Online. Avaiable from: https://doi.org/10.1016/B978-0-12-818366-3.00005-8 Accessed 18th October 2022.

[23] Google Developers. (2022). *'Classification: ROC Curve and AUC'*. Machine Learning. Foundational courses. Classification. Online. Available from: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc Accessed 18th October 2022.

[24] Brownlee, J. (2018). *'k-Fold Cross Validation'*. Statistical Methods for Machine Learning: Discover how to Transform Data into Knowledge with Python. N.p.: Machine Learning Mastery. Chapter 18 Estimation with Cross-Validation. p. 144.

[25] Yang PhD, S., Berdine MD, G. (2017) *'The receiver operating characteristic (ROC) curve'*. Online. Available from: doi:10.12746/swrccc.v5i19.391 Accessed 26th October 2022

[26] Pemberton-Roberts, L. (2022). *'SVC-Fine-Tuning.ipynb'*. Google Colab Pro. Online. Available from: https://colab.research.google.com/drive/1YWnjIZaYpS4wfmyo1rnDMNOmxKYcZ-M0?usp=sharing

[27] Pemberton-Roberts, L. (2022). *'LR-Fine-Tuning.ipynb'*. Google Colab Pro. Online. Available from: https://colab.research.google.com/drive/1D8i1HRDShhQAjLOs5PqpkeD0-FY9\_2Jm?usp=sharing

[28] Pemberton-Roberts, L. (2022). *'DT-Fine-Tuning.ipynb'*. Google Colab Pro. Online. Available from: https://colab.research.google.com/drive/1y1y09HvIFpJvek6ZxaQngODfwbE8eUPr?usp=sharing

[29] Joseph, R. (2018). *'Grid Search for model tuning'*. TDS. Medium. December 29th. Online. Available from: https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e Accessed 18th October 2022

[30] Bergstra, J., Bengio, Y. (2012). *'Random Search for Hyper-Parameter Optimization'*. Journal of Machine Learning Research, c13, pp 281-305. Online. Available from: https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf Accessed 18th October 2022.