

6006CEM Machine Learning and Related Application

<https://livecoventryac-my.sharepoint.com/:f:/r/personal/yeg3_uni_coventry_ac_uk/Documents/15490111-GRY-s1?csf=1&web=1&e=AF4nKV>

Name: GeRui Ye

Student ID: 15490111

Submission date: 3/12/2024

Course Information: 6006CEM Machine Learning

**Contents**

[1](#_Toc16351)**[. Abstract (Task 1)](#_Toc16351)** [1](#_Toc16351)

[1.1. Introduction 1](#_Toc8787)

[1.2. Objectives 1](#_Toc14258)

[1.3. Importance 1](#_Toc21974)

[2. Application of Machine Learning (Task 1) 1](#_Toc25430)

[2.1. Overview of the Task 1](#_Toc15101)

[2.2. Challenges and Future Directions 2](#_Toc19392)

[2.3. Image Captioning Methods 2](#_Toc18883)

[2.4. Conclusion 3](#_Toc6280)

[3](#_Toc1209)**[. Technical Implementation (Task 2)](#_Toc1209)** [4](#_Toc1209)

[3.1. Problem Specification 4](#_Toc24023)

[3.2. Comparative Analysis of Existing Work 4](#_Toc17373)

[3.3. Data Analysis and Preprocessing 5](#_Toc7770)

[3.4. Model Development 6](#_Toc1831)

[3.5. Model Optimization 7](#_Toc18979)

[3.6. Model Evaluation 7](#_Toc5745)

[3.7. Inclusion 11](#_Toc2418)

[4](#_Toc8611)**[. References](#_Toc8611)** [14](#_Toc8611)

[5](#_Toc23205)**[. Appendix A](#_Toc23205)** [15](#_Toc23205)

[6](#_Toc27472)**[. Appendix B](#_Toc27472)** [15](#_Toc27472)

|  |
| --- |
| **Academic Report** |

1. **Abstract (Task 1)**
   1. Introduction

With the rapid advancement of artificial intelligence, immense convenience has been brought to humanity, making AI development a major trend in today’s society.Khan et al. (2023) stated that Artificial intelligence is about making robots think like humans. Therefore machine learning applied to natural language processing, computer vision, to perform image processing is an important topic and the first step to make machines think like human beings.

* 1. Objectives

In Computer Vision (CV), ML plays a crucial role in extracting important information from images.Computer vision has made significant contributions in the fields of regulation, character recognition, robotics, and medicine, helping humans to extract image features. Research in computerised natural language processing aims to make language models more accurate and fluent, the first part will introduce the applications and challenges of CV and computerised language processing based on the research literature.

* 1. Importance

I will discuss Image Caption Generation which is a multi-scientific task that requires understanding of the visual scene, contextual elements and generating textual descriptions. Along the way, we will identify the challenges and limitations of these tasks, so that we can continue to discuss the effective use of machine learning in the direction of computer vision and natural language processing.Meanwhile, It will also recognise the evaluation metrics of the image description generation model, including: BLEU, METEOR, ROUGE and CIDEr. Derkar et al.(2021)used these metrics to derive model evaluation results for CaptionGenX.

1. **Application of Machine Learning (Task 1)**
   1. Overview of the Task

Machine learning (ML) has become an indispensable tool in the fields of natural language processing (NLP) and computer vision (CV),Therefore, computers need to deal with huge amounts of data, and how to handle the model correctly and adjust the ‘parameters’ is the key to success; One particularly intriguing application within computer vision is image captioning. The task combines elements of CV and NLP, involving the automatic generation of descriptive text based on the contents of an image.This literature review delves into the machine learning methods and models that have been employed to tackle image captioning, discussing their development, strengths, limitations, and future directions

* 1. Challenges and Future Directions

Take an example: for a machine, a very diﬀicult problem for humans, such as 928471 \* 938532, the machine can be completed in a very short time, but for very simple problems for humans, such as a 0-9 handwritten numbers, humans can immediately determine what is, but let the machine to determine the handwritten numbers are actually what is, it’s very diﬀicult; other case: a yellow object, let the ma- chine determine whether it is a small yellow dog, or hamburger, but for the machine is particularly diﬀicult.

In concrete terms, Derkar et al. mention that “Studying images requires, through algorithms, delving into pixel-level information, be deciphers the subtle nuances of shapes, colours, textures, and spatial relationships”. And often the real world contains a variety of domains, styles, and themes; it also has to rely on a complex software ecosystem.

However, Image description generation is of great scientific importance, which requiring an understanding of the visual scene, contextual elements, and the generation of fluent and relevant descriptionsIts applications include accessibility improvements for visually impaired individuals, content-based image retrieval, and improved human-computer interaction.This is of great research significance and challenge, firstly the image has to be recognised and processed, and then the processed image information has to be edited for text, features extracted, contextualised and understood to generate coherent and harmonious language.

* 1. Image Captioning Methods

CaptionGenX is a significant image captioning software at present. As noted by Derkar et al., CaptionGenX serves dual purposes. It is applicable in both computer vision and text processing fields, thus giving it the synergy of two. It has capabilities like multimedia analysis, content indexing, content extraction, and assistive technologies. In this case, a capable programming environment is critical for the segments of software development and its implementation. Libraries like Pandas and NLTK are therefore valuable since they furnish the primary functions for organizing databases and preparing text for analysis. The methodology includes a series of actions that utilize machine learning to ingest images and create informative captions.This method is illustrated in Flowchart (1).

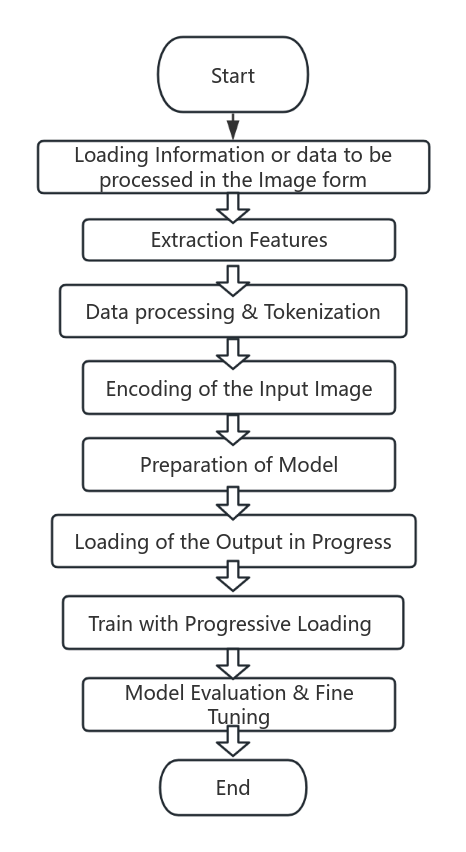


Fig. 1.

Showing flow chart for working of the CaptionGenX

The practical application is: given a picture of a puppy. The title prediction gives that the puppy is running in the snow. In addition Derkar et al(2023) given that Tabular representation of the evaluation of results of the model:

|  |  |
| --- | --- |
| BLEU Score (1-5) | 4.2 |
| METEOR Score (0-1) | 0.75 |
| CIDEr Score (0-inf) | 3.8 |

Table I. Tabular representation of the evaluation of results of the model

The score for the CaptionGenX model, achieving a BLEU of 4.2, signifies a strong correlation with the input subtitles, based on the evaluation measures. The evaluation measures assigned a METEOR score of 0.75, which shows a decent level of precision, recall, and agreement. A CIDEr score of 3.8 for the result indicates diversity, proving that the model is capable of capturing different characteristics of the photos.

* 1. Conclusion

Image description research in the future will be concentrated on model interpretability, multimodal learning to alleviate data bias, and the quality of human-like descriptions. VLP (Vision-Language Pretraining) is also underway. The influences of machine learning and computer vision on image description systems, including semantic differences and data bias evaluation. This may lead to more precise and diverse description systems. Ultimately, revealing visualizations from these evolving models will facilitate their impact on practical applications. Thus, the development will witness a higher level of interaction among humans and machine learning in diverse areas.

1. **Technical Implementation (Task 2)**
   1. Problem Specification

I collected a set of bank data about a customer, and whether or not the final customer purchased a time deposit; after processing it, there were 21 columns and 41189 rows of data from Moro et al. Produced by Bank Marketing [Dataset] from UCI Machine Learning Repository; This data contains information about the user’s age, job, education, housing, date of enquiry, etc.; from this data, a machine learning model is constructed to determine whether the final user will buy a fixed-term financial plan or not.

My Forecast can help banks and securities markets to predict user purchases, helping them to make decisions and marketing strategies; for customers, the data can be used to determine what regular products are suitable for them, market trends and provide good financial advice. The challenge also lies in analysing the customer's background information, economic indicators and many other factors to identify the customers most likely to subscribe, thus reducing the cost of the campaign and increasing its efficiency.

Technically, this is a machine learning classification problem because the target vari- able is a binary classification: Users decide whether they will buy term financial products (yes or no). By learning the historical data from the Information on the customer’s shop- ping history and personal background, we can train a model to predict the customer’ behaviour, which is typically a classification task in supervised learning.

* 1. Comparative Analysis of Existing Work

Taking into account the evolution of finance and technology, individuals tend to increasingly apply machine learning in financial markets to anticipate stock price movements, Bitcoin price changes, or optimize bank marketing activities. For example, Chen et al.(2020) were the ones who made predictions about Bitcoin price. The authors argue that machine learning algorithms like Random Forest, XGBoost, Quadratic Discriminant Analysis, Support Vector Machines, and LSTM are more efficient in predicting Bitcoin price at 5-minute intervals than statistical techniques.

Another paper by Setiyani et al (2022). examines the behaviour of bank users, using a variety of models to make predictions, and achieves a fairly high level of accuracy, but the technical aspects, such as how it is implemented and how the data is processed, are not discussed in depth. In my approach, several models are compared with each other. Methods for data processing and data optimisation are given. Finally, I provide data visualisations that make it possible to show the relevance of the data in an intuitive way.

* 1. Data Analysis and Preprocessing

In the data processing stage, I first used .info() and .isnull().sum() to check data types and missing values. I found 11 object-type columns, with no missing entries. So, I proceeded to analyze each column's characteristics by applying .value\_counts(). Based on the unique patterns in each column, I'll then apply suitable encoding methods, such as frequency encoding, label encoding, or target encoding.

The data types are shown in Figure 2:

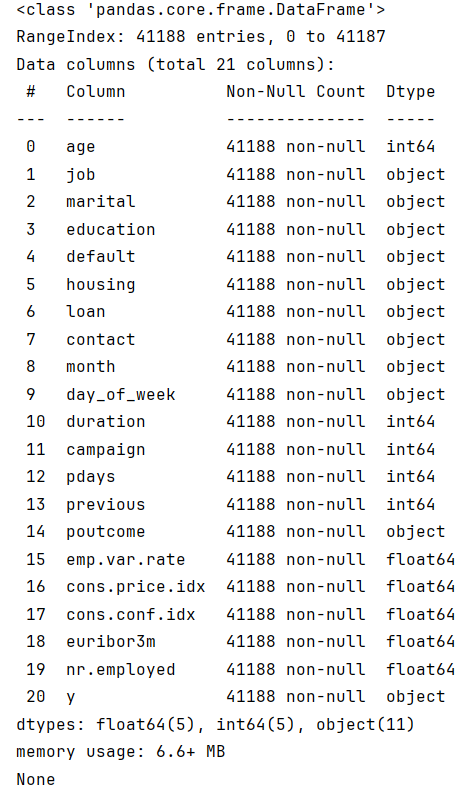


Figure 2

* The *job* column has a large number of unique values, with two values appearing infrequently, so frequency encoding is applied to maintain data usability.
* The marital column has four distinct values, with "unknown" being a very small portion, so frequency encoding is also used.
* The education column shows an ordinal pattern, from high to low education levels, so label encoding is applied to retain this order.
* The default column contains very few values, which could lead to overfitting, so frequency encoding is chosen.
* The default column contains very few values, which could lead to overfitting, so frequency encoding is chosen.
* The day\_of\_week and contact columns lack ordinal relationships and have weak target correlations, so frequency encoding is applied.

The preview after data processing is shown in Figure 3

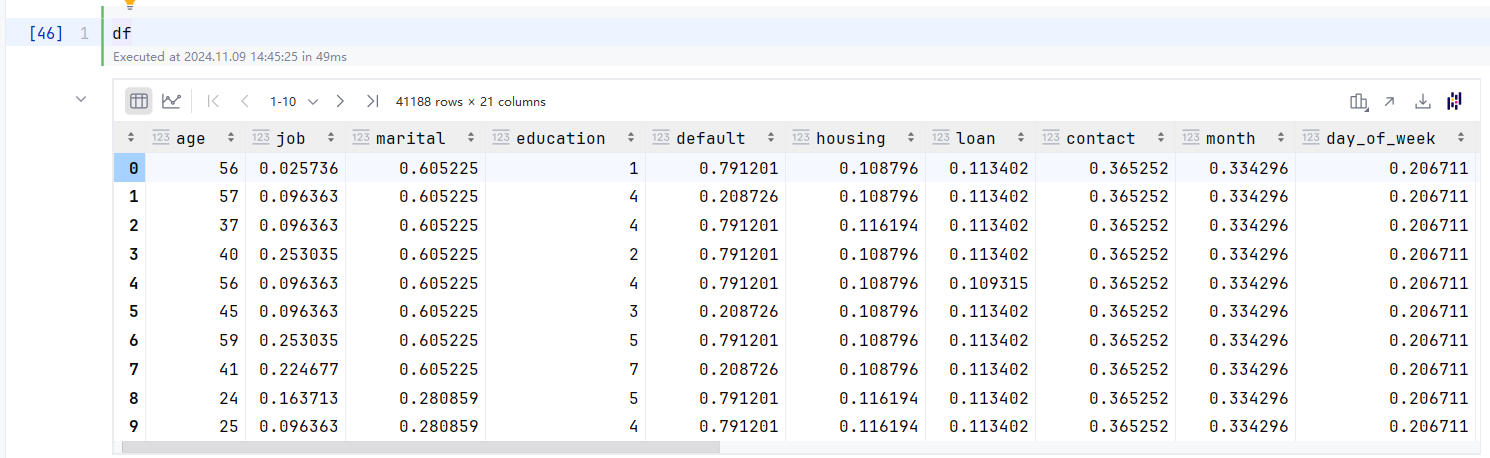


Figure 3

Next, I split the data into training and testing sets. First, I label purchasing users as Y and non-purchasing users as X. Given the low number of purchasing users, I use SMOTE to balance the classes, setting sampling\_strategy to 0.6 to maintain a modest sample size for the minority class. I set test\_size to 0.25, allocating 75% of the data for training and 25% for testing.

I also tested MinMaxScaler, StandardScaler, RobustScaler, and Normalizer to compare their effects on the scores of the test and training sets across different models. The closer the scores, the more suitable the scaler is for preprocessing. Based on the test results, StandardScaler and RobustScaler were chosen as the main preprocessing methods, Assigning values to test and training sets. The scores of MinMaxScaler, StandardScaler, RobustScaler, Normalizer are shown in Figure 4.

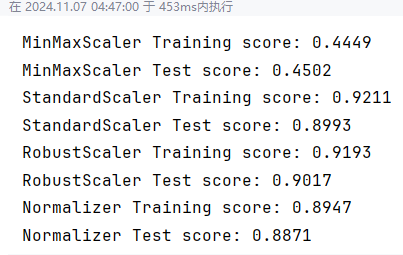


Figure 4

* 1. Model Development

I primarily studied three commonly used models and also built other existing algorithm models for comparison. Below is the practical application process of building these models:

When training the logistic regression algorithm, I set the maximum iterations (max\_iter) to 10,000 and the regularization parameter (C) to 1 to control sensitivity to noise. The solver is set to the default lbfgs, suitable for binary classification tasks. Then, I use .fit to optimize model parameters and generate the final model.

In the Neural Network model, I use a Multilayer Perceptron (MLP), inspired by human brain neurons, to address this problem. The code hidden\_layer\_sizes=(100,50) creates the first layer with 100 neurons and the second with 50 neurons, while random\_state sets a random seed to ensure reproducibility. The model is then generated accordingly.

In the KNN model, I utilized the dataset processed with StandardScaler, setting the number of neighbors (n\_neighbors) to 5 with weights set to 'uniform'. The KNN model was then applied to the standardized dataset features.

In addition to the three common models mentioned above, I also used RandomForestClassifier to build a random forest model, XGBClassifier to construct an XGBoost model, and LGBMClassifier for a LightGBM model, and the SVM. These models were then evaluated together.

* 1. Model Optimization

To enhance model performance, I first applied StandardScaler and RobustScaler as part of the scaling process. The goal was to achieve better performance and efficiency of the models. Nevertheless, I found that there was a distribution of the samples that were not balanced.

In the initial model training, though there was a high accuracy, precision and recall, as well as F1 score, fell short, the values were all about 60%. To tackle this, I used the SMOTE method to strike a balance in the model and restrict the overfitting effect.

On the other hand, I utilized cross-validation for assessing the model performance and the combination of parameters which outperformed others. The GridSearchCV took ages, hence I opted for RandomizedSearchCV, which samples a few.

I specified the ranges and weights for n\_neighbors and weights, respectively. Then I imported the model with the estimator and set the cross-validation parameter cv=5. Consequently, I allocated the resulting model to y\_pred. All metrics showed improvement after the evaluation of the outlined results.

* 1. Model Evaluation

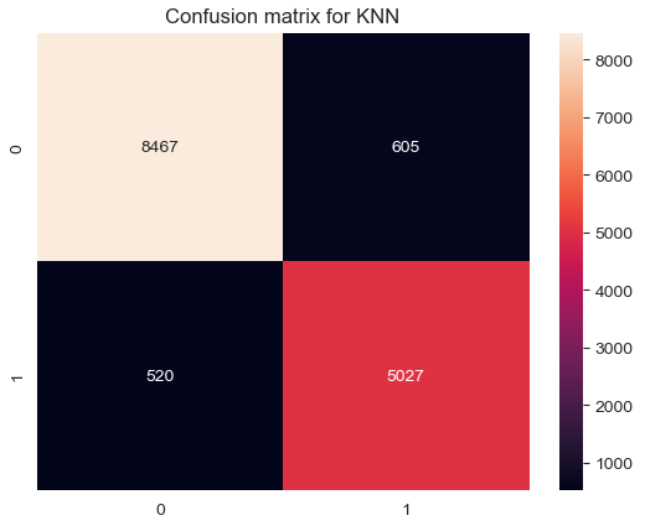
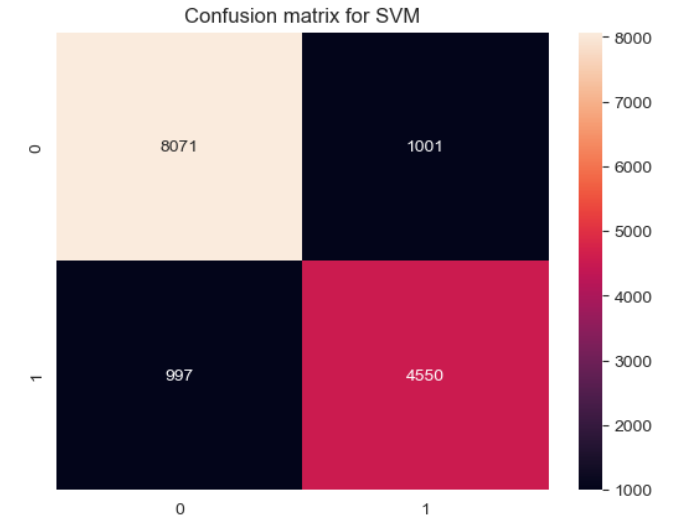
I used the following metrics to evaluate the model, The final evaluation results and performance of all models are shown in Table 2.:

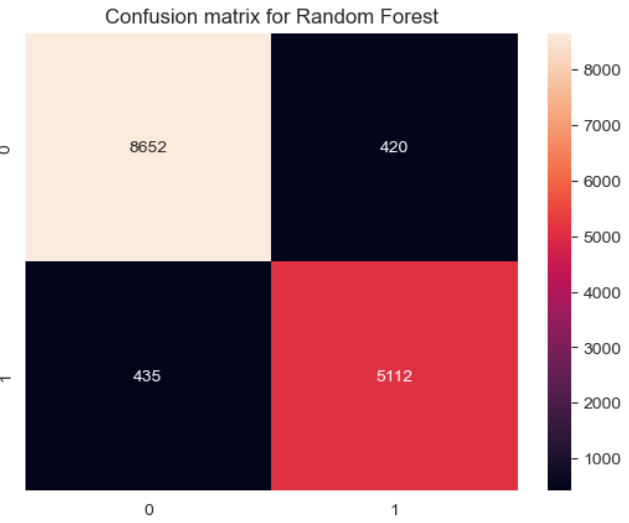
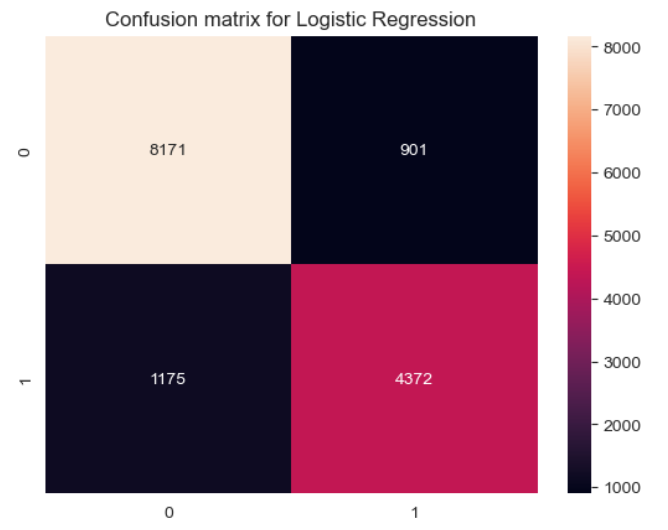
* Model Accuracy
* Model Precision
* Model Recall
* Model F1 Score
* Confusion Matrix
* ROC AUC plots

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Models** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| Logistic Regression | 0.8580 | 0.8291 | 0.7882 | 0.8081 |
| Neural Network | 0.9230 | 0.8926 | 0.9063 | 0.8994 |
| KNN | 0.9276 | 0.8811 | 0.9353 | 0.9074 |
| SVM  Random Forest  XGBoost  LightGBM | 0.8633  0.9415  0.9373  0.9415 | 0.8197  0.9241  0.9249  0.9282 | 0.8203  0.9216  0.9086  0.9158 | 0.8200  0.9228  0.9167  0.9220 |

Table 2

As shown in the figure, the confusion matrix diagram of the model is shown below:





I compared these five charts and concluded that Random Forest has the lowest number of false positives (FP) and false negatives (FN), meaning its error rate during testing is the smallest.

In contrast, Logistic Regression and SVM show higher FP and FN counts, indicating a higher error rate in comparison.

Additionally, Random Forest and KNN have the highest true positives (TP) and true negatives (TN), demonstrating the strongest ability to correctly predict both positive and negative cases.

The comparison of the ROC curves for developed model showing in Fig. 5.

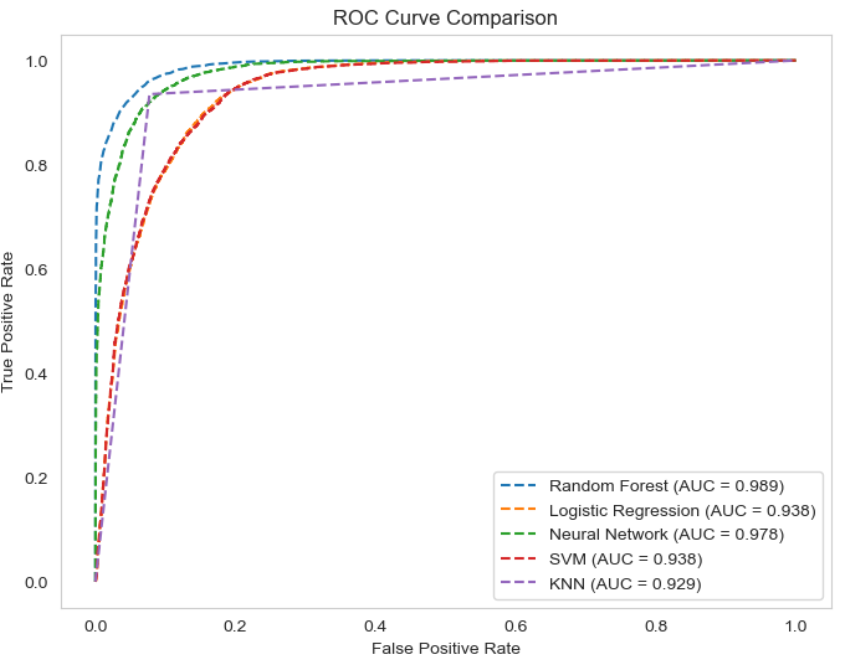


Fig 5.

Figure 6 displays a heatmap, highlighting strong correlations concentrated in the lower-right section (shown in shades of red).

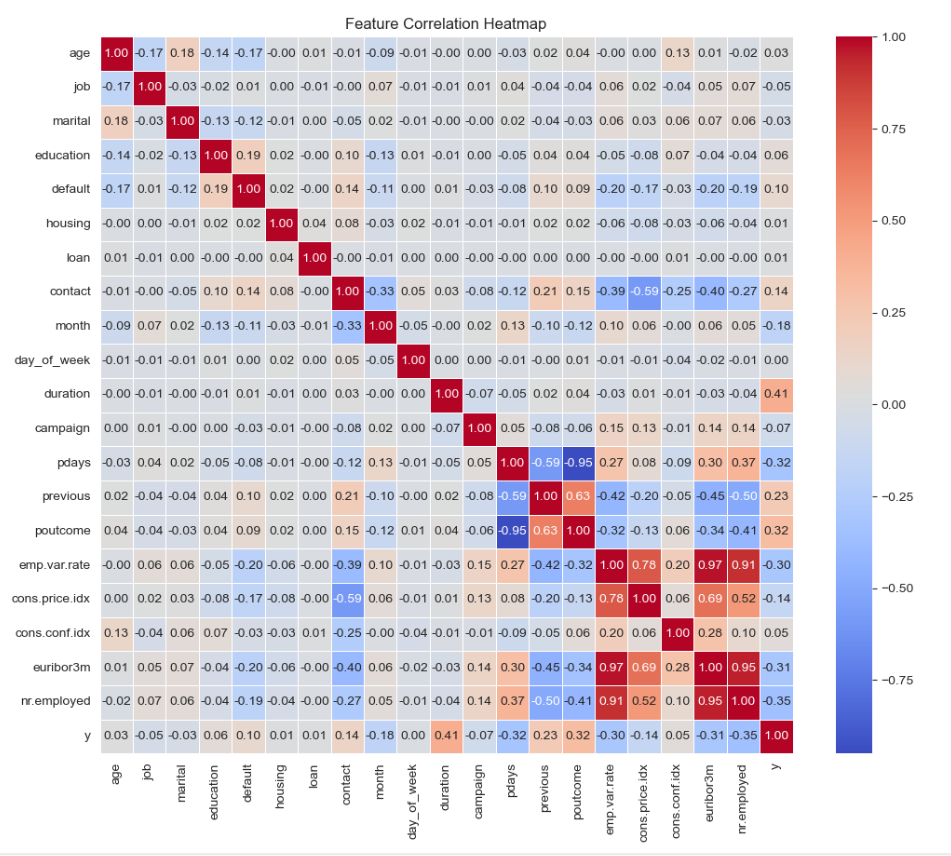
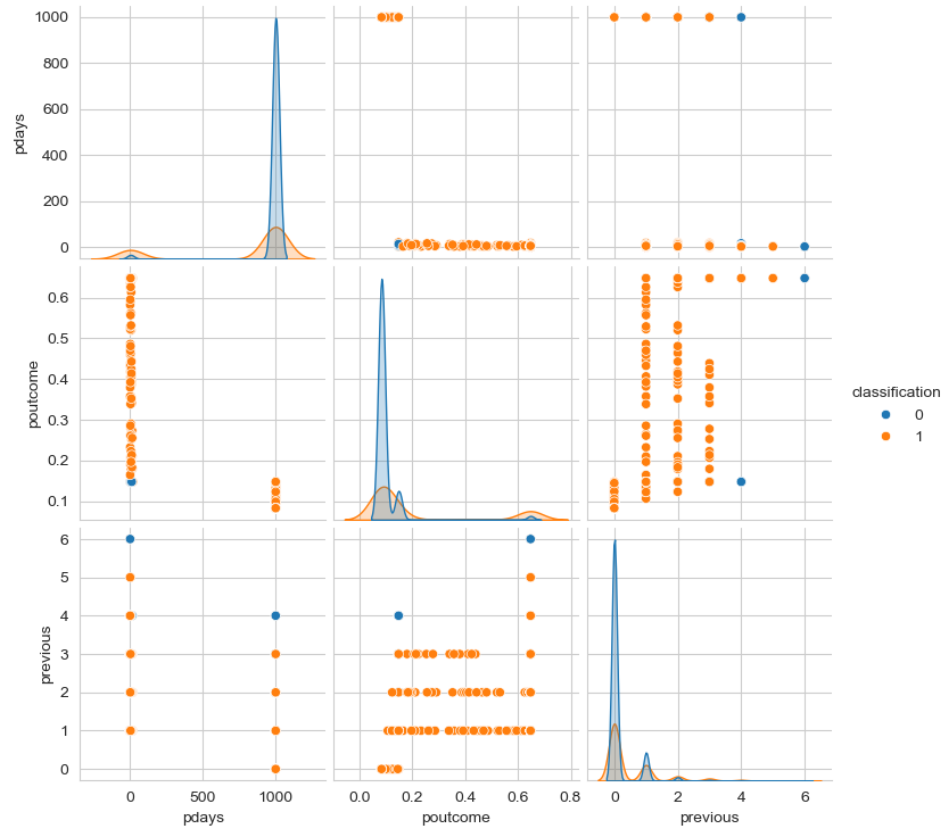


Fig 6.

Being based on this heatmap, here is an analysis of the most representative correlations:

* emp.var.rate positively correlated with cons.conf.idx: The correlation coefficient of 0.78 demonstrates that a higher Employment Variation Rate is linked to a higher Consumer Confidence Index.
* emp.var.rate positively correlated with euribor3m: As a result, this data point reflects a strong positive correlation, with a coefficient of 0.97 between the Employment Variation Rate and the Euribor 3-month interest rate.
* emp.var.rate positively correlated with nr.employed: Therefore, the high positive correlation value of 0.91 illustrates that the increase in Employment Variation Rate is directly connected to more employees being hired.
* nr.employed positively correlated with euribor3m: A very high correlation of 0.95 implies that the number of employees has consistency with a higher value of euribor for three months.
* There’s a strong negative correlation (-0.95) between pdays and poutcome, indicating that as the Days Since Previous Contact increases, the Outcome of the Previous Campaign tends to decrease.
* There’s a moderate negative correlation (-0.59) between previous and pdays, suggesting that higher values of Days Since Previous Contact are associated with fewer interactions in the previous campaign.
* A moderate positive correlation (0.63) exists between previous and poutcome, meaning the higher the Outcome of the Previous Campaign, the more interactions occurred during that campaign.
* Correlations among the remaining variables are relatively weak, mostly falling below 0.5. The remaining relationships are not very strong and go mostly below this mark.
  1. Inclusion

According to the model performance in Table 2 and the observations inferred from the ROC curves, Random Forest obtained the best performance, while the Logistic Regression model fell short of the other models. For Setiyani et al., the XGBoost model appeared to have the best performance, which was around 91%, while my trial runs for Random Forest came close to 94% with all the metrics. Future-oriented, more intensive deep learning data along with outside data will improve the model even more.The figure shows a pairplot plot with several columns that are negatively correlated:



From the pairplot, we can draw the following conclusions:

* In the pdays plot, values around 0 tend to cluster near 1000, indicating that customers who have never been contacted before are less likely to make a purchase.
* In the poutcome plot, values near 0 are concentrated around 0.1, suggesting that customers are unlikely to buy again if the previous marketing attempt was unsuccessful.
* In the previous plot, values near 0 are mostly at 0, implying that the fewer times a customer was previously contacted, the lower the likelihood of them purchasing the product.
* Based on the scatter plot, the distributions of pdays, poutcome, and previous stand out clearly, showing independent patterns.

The figure shows a pairplot of several columns that are positively correlated:

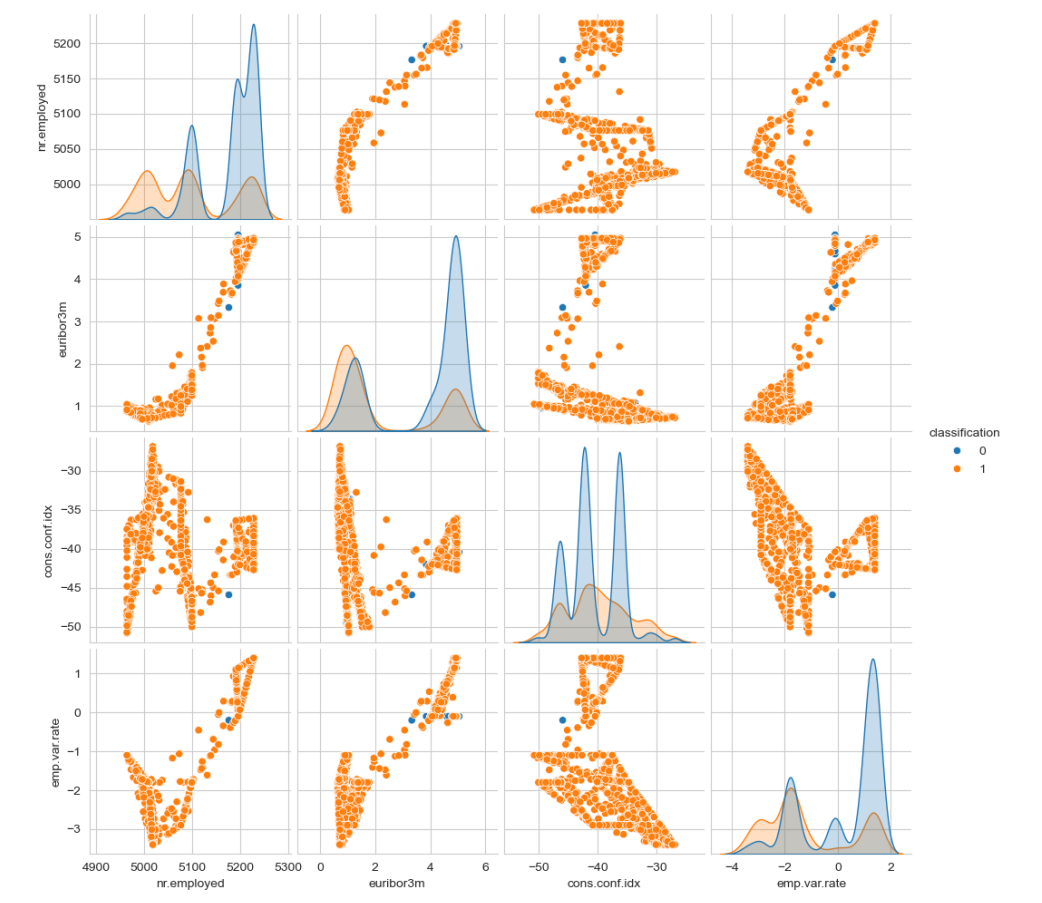


Fig.

* In the nr.employed chart, we can see that the "0" values mostly cluster around the 5000 mark. This indicates that as the total number of employed individuals increases, customers are less likely to purchase financial products.
* In the euribor3m chart, "0" values are more concentrated near 3.5, suggesting that customers tend not to buy financial products when the interest rate is at 3.5.
* Looking at the cons.conf.idx chart, the "0" values appear more scattered, while "1" values are mostly concentrated around -38. This implies that customers are more likely to buy financial products when consumer confidence is at a moderate level.
* In the emp.var.rate chart, "0" values are heavily concentrated around 1. This suggests that customers tend to avoid buying financial products during periods of employment growth.
* A scatter plot reveals that the distribution of 0s and 1s in these four columns isn't distinctly separated, with some overlap present. Users inclined to purchase financial products tend to be more widely spread out.

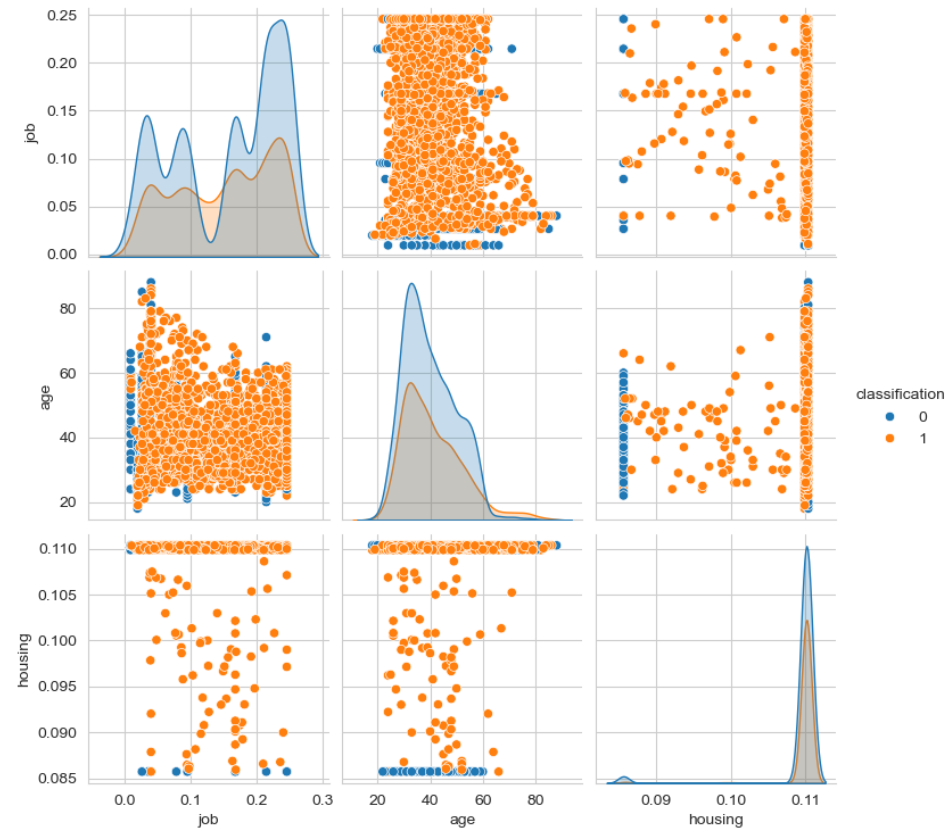


Fig.

* I also included a few columns with lower relevance: from the diagonal plot, it’s clear that factors like job, housing, and age don’t play a significant role in determining whether a customer chooses to buy financial products.

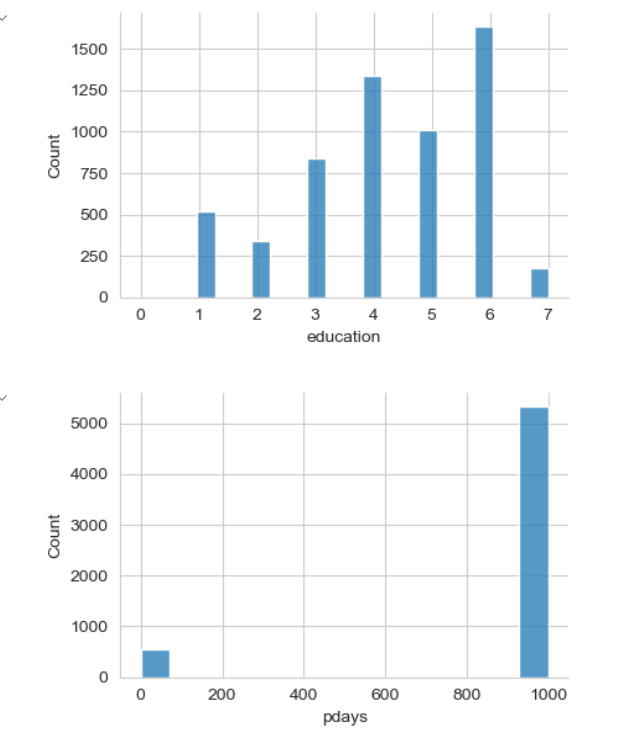
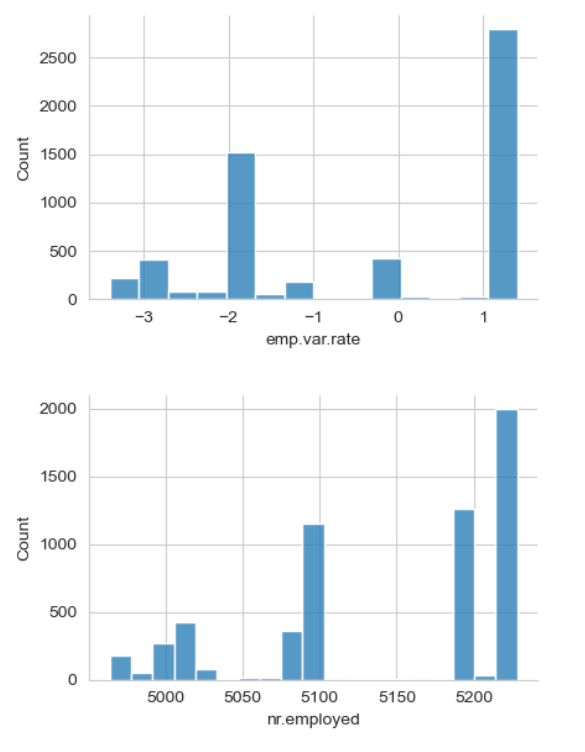
|  |
| --- |
| **Bibliography** |

1. **References**
2. Khan, A., Laghari, A., & Awan, S. (2018). Machine Learning in Computer Vision: A Review. EAI Endorsed Trans. Scalable Inf. Syst., 8, e4. <https://doi.org/10.4108/EAI.21-4-2021.169418.>
3. Derkar, S. B., Biranje, D., Thakare, L. P., Paraskar, S., & Agrawal, R. (2023). CaptionGenX: Advancements in deep learning for automated image captioning. Proceedings of the 3rd Asian Conference on Innovation in Technology (ASIANCON), Ravet, India, 1-8. <https://doi.org/10.1109/ASIANCON58793.2023.10270020>
4. Chen, Z., Li, C., & Sun, W. (2020). Bitcoin price prediction using machine learning: An approach to sample dimension engineering. Journal of Computational and Applied Mathematics, 365, 112395. [https://doi.org/10.1016/j.cam.2019.112395](https://doi.org/10.1016/j.cam.2019.112395" \t "_new)
5. Ortiz-Garces, I.; Govea, J.; Andrade, R.O.; Villegas-Ch, W. Optimizing Chatbot Effectiveness through Advanced Syntactic Analysis: A Comprehen- sive Study in Natural Language Processing. Appl. Sci. 2024, 14(5), 1737. <https://doi.org/10.3390/app14051737>
6. Kang, Y.; Cai, Z.; Tan, C.W.; Huang, Q.; Liu, H. Natural Language Processing (NLP) in Management Research: A Literature Review. J. Manag. Anal. 2020, 7, 139–172
7. Netzer, Yuval, Tao Wang, Adam Coates, AlessandroBissacco, Bo Wu, and Andrew Y. Ng. ”Reading digits in natural images with unsupervised feature learning.” (2011).
8. Zhao, Ying, and Arjuna Flenner. ”Deep Models, Machine Learning, and Artificial Intelligence Applications in National and International Security.” AI Magazine 40, no. 1 (2019): 35-36..
9. Setiyani, L., Indahsari, A., Rosalina, & Wansen, T. (2022). Finding the best techniques for predicting term deposit subscriptions (Case study UCI machine learning dataset). 2022 IEEE International Conference on Sustainable Engineering and Creative Computing (ICSECC), Kabupaten Bekasi, Indonesia, 100-104. <https://doi.org/10.1109/ICSECC56055.2022.10331379>
10. Moro, S., Rita, P., & Cortez, P. (2014). Bank Marketing [Dataset]. UCI Machine Learning Repository. <https://doi.org/10.24432/C5K306.>
11. Khajuria, D., Sharma, A., Sharma, N., & Mangla, M. (2023). Classification and comparative analysis of Earth's nearest objects using machine learning models. 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom), 16–23. New Delhi, India.

|  |
| --- |
| **Appendix A** |

1. **Appendix A**

*The following image is a data visualization of some columns:*



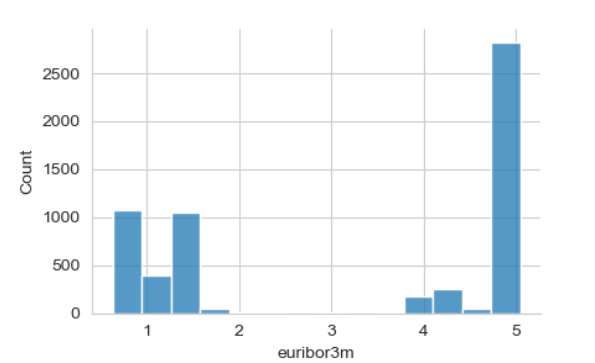
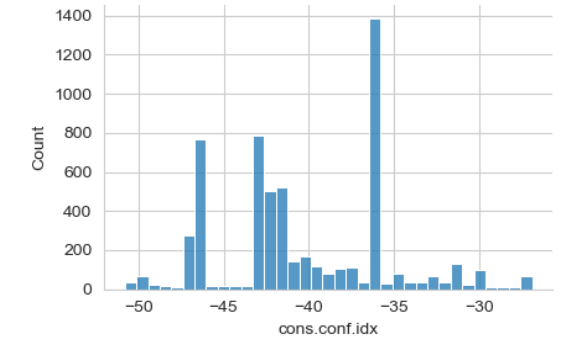


Fig.

|  |
| --- |
| **Appendix B** |

1. **Appendix B**

Programming code: Iused “highlightcode.com” to convert code into highlighted text format, with each code section separated by #====== for easier reference.

1. import pandas as pd
2. import numpy as np
3. # Random Seed
4. # classification
5. file\_path = 'bank-additional-full.csv'
6. df = pd.read\_csv(file\_path, sep=';')
7. df.shape
8. df
9. # ================================
10. df.isnull().sum()
11. # There are no missing values in the data
12. # check the shape of the data
13. df.info()
14. # ================================
15. print(df['y'].value\_counts())
16. # Firstly, column y is treated as 0, 1 by treating whether people will buy time deposits or not
17. df['y']=df['y'].map({'yes':1,'no':0})
18. # ================================
19. print(df['job'].value\_counts())
20. # After entering the results, it was found that there were more values in the work, but two less data,
21. # so frequency coding was chosen
22. job\_freq = df['job'].value\_counts()/len(df)
23. df['job']=df['job'].map(job\_freq)
24. # ================================
25. print(df['marital'].value\_counts())
26. # It was found that there are four values in marital and the percentage of unknown is small,
27. # and there is no order relationship so the frequency code was chosen.
28. marital\_freq = df['marital'].value\_counts() / len(df)
29. df['marital'] = df['marital'].map(marital\_freq)
30. # ================================
31. print(df['education'].value\_counts())
32. # A sequential relationship was found to exist for EDUCATION, so label coding was used
33. education\_order = ['illiterate', 'basic.4y', 'basic.6y', 'basic.9y',
34. 'high.school', 'professional.course', 'university.degree', 'unknown']
35. df['education']=df['education'].apply(lambda x: education\_order.index(x))
36. # ================================
37. print(df['education'].value\_counts())
38. print(df['default'].value\_counts())
39. # This column was found to have a small target term,
40. # but YES the target was very small, so the target coding was easily overfitted,
41. # so frequency coding was used
42. default\_freq = df['default'].value\_counts()/len(df)
43. df['default']=df['default'].map(default\_freq)
44. # ================================
45. print(df['housing'].value\_counts())
46. # Appears UNKNOWN and the number is small so it is adopted, so use the target code
47. target\_housing=df.groupby('housing')['y'].mean()
48. df['housing']=df['housing'].map(target\_housing)
49. # ================================
50. print(df['loan'].value\_counts())
51. # same with housing, use the target code
52. target\_loan=df.groupby('loan')['y'].mean()
53. df['loan']=df['loan'].map(target\_loan)
54. # ================================
55. print(df['contact'].value\_counts())
56. # Smaller quantities, frequency coded
57. contact\_freq = df['contact'].value\_counts()/len(df)
58. df['contact']=df['contact'].map(contact\_freq)
59. # ================================
60. print(df['month'].value\_counts())
61. month\_freq=df['month'].value\_counts()/len(df)
62. df['month']=df['month'].map(month\_freq)
63. # ================================
64. print(df['day\_of\_week'].value\_counts())
65. #same with month, use the target code
66. week\_freq=df['day\_of\_week'].value\_counts()/len(df)
67. df['day\_of\_week']=df['day\_of\_week'].map(week\_freq)
68. # ================================
69. print(df['poutcome'].value\_counts())
70. # Relationships may exist with whether or not they will be purchased on a regular basis,
71. # so target coding is used
72. target\_poutcome=df.groupby('poutcome')['y'].mean()
73. df['poutcome']=df['poutcome'].map(target\_poutcome)
74. # ================================
75. df.info()
76. df
77. from sklearn.model\_selection import train\_test\_split
78. from imblearn.over\_sampling import SMOTE
79. X = df.drop(columns=['y'])
80. y = df['y']
81. # Initial observations suggested that the algorithm was justified, but the precision was disappointing. In order to overcome the data inbalance, the SMOTE technique is used.
82. smote = SMOTE(random\_state=42, sampling\_strategy=0.6)
83. X\_resampled, y\_resampled = smote.fit\_resample(X, y)
84. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_resampled, y\_resampled, test\_size=0.25, random\_state=42)
85. # =================================
86. from sklearn.preprocessing import StandardScaler
87. scaler = StandardScaler()
88. X\_train = scaler.fit\_transform(X\_train)
89. X\_test = scaler.transform(X\_test)
90. # =================================
91. from sklearn.ensemble import RandomForestClassifier
92. model\_rf = RandomForestClassifier(n\_estimators=100, random\_state=42)
93. model\_rf.fit(X\_train, y\_train)
94. train\_score = model\_rf.score(X\_train, y\_train)
95. test\_score = model\_rf.score(X\_test, y\_test)
96. print(f"Training Score: {train\_score:.4f}")
97. print(f"Test Score: {test\_score:.4f}")
98. # =================================
99. from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score
100. y\_pred\_rf = model\_rf.predict(X\_test)
101. accuracy = accuracy\_score(y\_test, y\_pred\_rf)
102. precision = precision\_score(y\_test, y\_pred\_rf)
103. recall = recall\_score(y\_test, y\_pred\_rf)
104. f1 = f1\_score(y\_test, y\_pred\_rf)
105. print(f"Random Forest\_Model Accuracy:, {accuracy:.4f}")
106. print(f"Random Forest\_Model Precision: {precision:.4f}")
107. print(f"Random Forest\_Model Recall: {recall:.4f}")
108. print(f"Random Forest\_Model F1 Score: {f1:.4f}")
109. from sklearn.metrics import confusion\_matrix, classification\_report
110. cm = confusion\_matrix(y\_test, y\_pred\_rf)
111. print("Random Forest Confusion Matrix:\n", cm)
112. sns.heatmap(cm, annot=True, fmt="d").set\_title("Confusion matrix for Random Forest")
113. # ================================
114. # Logistic Regression Model
115. from sklearn.linear\_model import LogisticRegression
116. # Create a logistic regression model, set the maximum number of iterations to 10000
117. model\_lr = LogisticRegression(max\_iter=10000, C=1, solver='lbfgs')
118. model\_lr.fit(X\_train, y\_train)
119. from sklearn.metrics import accuracy\_score
120. from sklearn.metrics import precision\_score, recall\_score, f1\_score
121. # Predict the results for the test set
122. pred\_lr = model\_lr.predict(X\_test)
123. accuracy = accuracy\_score(y\_test, y\_pred\_lr)
124. # Calculate the precision, recall, and F1 score
125. precision = precision\_score(y\_test, y\_pred\_lr)
126. recall = recall\_score(y\_test, y\_pred\_lr)
127. f1 = f1\_score(y\_test, y\_pred\_lr)
128. # Display the accuracy, precision, recall, and F1 score
129. print(f"Logistic Regression Model Recall: {recall:.4f}")
130. print(f"Logistic Regression Model Accuracy: {accuracy:.4f}")
131. print(f"Logistic Regression Model Precision: {precision:.4f}")
132. print(f"Logistic Regression Model F1 Score: {f1:.4f}")
133. from sklearn.metrics import confusion\_matrix
134. # Calculate the confusion matrix
135. cm = confusion\_matrix(y\_test, y\_pred\_lr)
136. print("Logistic Regression Confusion Matrix:\n", cm)
137. sns.heatmap(cm, annot=True, fmt="d").set\_title("Confusion matrix for Logistic Regression")
138. # ================================
139. import MLPClassifier from sklearn.neural\_network
140. model\_MLP = MLPClassifier(hidden\_layer\_sizes=(100, 50), max\_iter=1000, random\_state=0)
141. model\_MLP.fit(X\_train, y\_train)
142. train\_score = model\_MLP.score(X\_train, y\_train)
143. test\_score = model\_MLP.score(X\_test, y\_test)
144. print(f"Training Score: {train\_score:.4f}")
145. print(f"Test Score: {test\_score:.4f}")
146. # =================================
147. from sklearn.metrics import accuracy\_score
148. from sklearn.metrics import precision\_score, recall\_score, f1\_score
149. y\_pred = model\_MLP.predict(X\_test)
150. accuracy = accuracy\_score(y\_test, y\_pred)
151. precision = precision\_score(y\_test, y\_pred)
152. recall = recall\_score(y\_test, y\_pred)
153. f1 = f1\_score(y\_test, y\_pred)
154. print(f"Neural Network Model Accuracy: {accuracy:.4f}")
155. print(f"Neural Network Model Precision: {precision:.4f}")
156. print(f"Neural Network Model Recall: {recall:.4f}")
157. print(f"Neural Network Model F1 Score: {f1:.4f}")
158. from sklearn.metrics import confusion\_matrix
159. cm = confusion\_matrix(y\_test, y\_pred)
160. print("Neural Network Confusion Matrix:\n", cm)
161. sns.heatmap(cm, annot=True, fmt="d").set\_title("Confusion matrix for Neural Network")
162. # =================================
163. from sklearn.svm import SVC
164. from sklearn.neighbors import KNeighborsClassifier
165. from sklearn.preprocessing import StandardScaler
166. # SVM with GridSearchCV
167. model\_svm = SVC(kernel='linear', C=5.0, random\_state=0)
168. # Test the value of C by setting it to a high and low value to get the best accuracy
169. model\_svm.fit(X\_train, y\_train)
170. # ================================
171. from sklearn.metrics import accuracy\_score
172. from sklearn.metrics import precision\_score, recall\_score, f1\_score
173. y\_pred = model\_svm.predict(X\_test)
174. accuracy = accuracy\_score(y\_test, y\_pred)
175. precision = precision\_score(y\_test, y\_pred)
176. recall = recall\_score(y\_test, y\_pred)
177. f1 = f1\_score(y\_test, y\_pred)
178. print(f"SVM Model Precision: {precision:.4f}")
179. print(f"SVM Model Accuracy: {accuracy:.4f}")
180. print(f"SVM Model Recall: {recall:.4f}")
181. print(f"SVM Model F1 Score: {f1:.4f}")
182. cm = confusion\_matrix(y\_test, y\_pred)
183. print("SVM Confusion Matrix:\n", cm)
184. sns.heatmap(cm, annot=True, fmt="d").set\_title("Confusion matrix for SVM")
185. # ================================
186. from sklearn.neighbors import KNeighborsClassifier
187. from sklearn.model\_selection import RandomizedSearchCV
188. from sklearn.metrics import classification\_report
189. # Define a range of NI parameters
190. param\_dist = {
191. 'n\_neighbors': np.arange(1, 31),
192. 'weights': ['uniform', 'distance']
193. }
194. model\_knn = KNeighborsClassifier(n\_neighbors=5, weights='uniform')
195. # Create the Randomized Search CV
196. random\_search = RandomizedSearchCV(
197. estimator=model\_knn,
198. param\_distributions=param\_dist,
199. n\_iter=20,
200. scoring='accuracy',
201. cv=5,
202. random\_state=42
203. )
204. # Start the random search process
205. random\_search.fit(X\_train, y\_train)
206. # Show best params in the Input Stage
207. print("Best parameters found:", random\_search.best\_params\_)
208. print("Best cross-validation score:", random\_search.best\_score\_)
209. # Make predictions with the best model
210. best\_knn = random\_search.best\_estimator\_
211. y\_pred = best\_knn.predict(X\_test)
212. #===============================
213. from sklearn.metrics import accuracy\_score
214. from sklearn.metrics import precision\_score, recall\_score, f1\_score
215. accuracy=accuracy\_score(y\_test, y\_pred)
216. precision = precision\_score(y\_test, y\_pred)
217. recall = recall\_score(y\_test, y\_pred)
218. f1 = f1\_score(y\_test, y\_pred)
219. print(f"KNN Model Accuracy:, {accuracy:.4f}")
220. print(f"KNN Model Precision: {precision:.4f}")
221. print(f"KNN Model Recall: {recall:.4f}")
222. print(f"KNN Model F1 Score: {f1:.4f}")
223. cm = confusion\_matrix(y\_test, y\_pred)
224. print("SVM Confusion Matrix:\n", cm)
225. sns.heatmap(cm, annot=True, fmt="d").set\_title("Confusion matrix for KNN")
226. #===============================
227. import matplotlib.pyplot as plt
228. from sklearn.metrics import roc\_curve, roc\_auc\_score
229. # ROC Curve for RandomForest
230. rf\_probs = model\_rf.predict\_proba(X\_test)[:, 1]
231. rf\_fpr, rf\_tpr, \_ = roc\_curve(y\_test, rf\_probs)
232. rf\_auc = roc\_auc\_score(y\_test, rf\_probs)
233. # ROC Curve for Logistic Regression
234. lr\_probs = model\_lr.predict\_proba(X\_test)[:, 1]
235. lr\_fpr, lr\_tpr, \_ = roc\_curve(y\_test, lr\_probs)
236. lr\_auc = roc\_auc\_score(y\_test, lr\_probs)
237. # ROC Curve for Neural Network
238. mlp\_probs = model\_MLP.predict\_proba(X\_test)[:, 1]
239. mlp\_fpr, mlp\_tpr, \_ = roc\_curve(y\_test, mlp\_probs)
240. mlp\_auc = roc\_auc\_score(y\_test, mlp\_probs)
241. # ROC Curve for SVM
242. svm\_probs = model\_svm.decision\_function(X\_test)
243. svm\_fpr, svm\_tpr, \_ = roc\_curve(y\_test, svm\_probs)
244. svm\_auc = roc\_auc\_score(y\_test, svm\_probs)
245. # ROC Curve for KNN
246. knn\_probs = best\_knn.predict\_proba(X\_test)[:, 1]
247. knn\_fpr, knn\_tpr, \_ = roc\_curve(y\_test, knn\_probs)
248. knn\_auc = roc\_auc\_score(y\_test, knn\_probs)
249. # Plotting all the ROC curves on the same plot
250. plt.figure(figsize=(8, 6))
251. plt.plot(rf\_fpr, rf\_tpr, linestyle='--', label=f'Random Forest (AUC={rf\_auc:.3f})')
252. plt.plot(lr\_fpr, lr\_tpr, linestyle='--', label=f'Logistic Regression (AUC={lr\_auc:.3f})')
253. plt.plot(mlp\_fpr, mlp\_tpr, linestyle='--', label=f'Neural Network (AUC={mlp\_auc:.3f})')
254. plt.plot(svm\_fpr, svm\_tpr, linestyle='--', label=f'SVM (AUC={svm\_auc:.3f})')
255. plt.plot(knn\_fpr, knn\_tpr, linestyle='--', label=f'KNN (AUC={knn\_auc:.3f})')
256. plt.xlabel('False Positive Rate')
257. plt.ylabel('True Positive Rate')
258. plt.title('ROC Curve Comparison')
259. plt.legend(loc='best')
260. plt.grid()
261. plt.show()
262. # =========================
263. import seaborn as sns
264. plt.figure(figsize=(12, 10))
265. sns.heatmap(df.corr().round(2), annot=True, cmap='coolwarm', linewidths=0.5, fmt='.2f')
266. plt.title('Feature Correlation Heatmap')
267. plt.show()
268. import pandas as pd
269. import seaborn as sns
270. import matplotlib.pyplot as plt
271. # Convert y\_resampled from Series to DataFrame
272. y\_resampled = pd.DataFrame(y\_resampled, columns=['y'])
273. # Plot the pairplot of selected features and y\_resampled
274. selected\_features = X\_resampled[['nr.employed', 'euribor3m', 'cons.conf.idx', 'emp.var.rate']]
275. combined\_data = pd.concat([selected\_features, y\_resampled], axis=1)
276. sns.pairplot(combined\_data, hue='y')
277. plt.show()