LISA Report - Project Jacopo Lazzari (894361)

Introduction

The **individual project** for the **Lab of Information Systems and Analytics** course consisted of applying Machine Learning to perform a predictive task starting from a given dataset to be chosen from 3 different datasets.

I chose to analyse the dataset related to the direct marketing campaigns of a Portuguese banking institution. These marketing campaigns were conducted via phone calls, where multiple contacts with the same client were often required to assess whether the client would subscribe to a term deposit (“yes”) or not (“no”).

The dataset is sourced from the UC Irvine archive (https://archive.ics.uci.edu/dataset/222/bank+marketing). There are 4 datasets available, I used the **bank-full.csv** one that contains all the examples and 17 inputs, ordered by date.

The **classification goal** was to predict whether the client would subscribe (yes/no) to a term deposit (target variable: `**y**`).

1. Dataset Loading and Cleaning

Firstly, I imported all the necessary **libraries** for my project: Pandas, NumPy, Matplotlib, Seaborn, Warnings, and the various functions and modules from the **scikit-learn** open-source Machine Learning library for Python.

In this section, the goal is to **load** the dataset and **perform** initial **data-cleaning** steps to prepare it for further analysis. The dataset, **bank-full.csv**, contains detailed information on client demographics, previous marketing interactions, and various economic indicators. So the task is to handle any inconsistencies or missing values, ensuring that the data is ready for feature engineering and model training.

Upon loading the dataset, I first examined the structure and contents to get a sense of the data. The dataset had **41,188 rows** and **21 columns**. I explored the data types and the first few rows to better understand it.

Initially, I observed that there were some “unknown” values and that the dataset included both numerical and categorical variables. I counted the number of “unknown” values in each column and found out that there were many. I replaced these “unknown” values with **`NaN`** to ensure the accuracy and reliability of the models I would build.

I studied the distribution of NaNs across columns and rows and observed that some columns had higher numbers of NaNs than others. **One-quarter** of all **rows** in the dataset **had NaNs**, but the distribution of outcomes for the target variable y was relatively consistent: 7.3% yes with NaNs and 12.7% without NaNs, compared to the total 11.3% yes for the entire dataset.

I examined the **correlation** between columns with NaNs to determine whether there were any consistent relationships between missing values in different columns but found that the correlation was generally neutral. Thus, I decided to **drop** the rows with **NaN values**.
Even by dropping around one-quarter of the total data, I believe this approach best preserves data integrity, avoids biases from imputation, and ensures reliable analysis and model performance later on. Moreover, with 30,488 rows remaining out of 41,188, the dataset is still large enough for our purposes.

2. Exploratory Data Analysis (EDA)

In the Exploratory Data Analysis section, the aim is to gain a better understanding of the data, drop or transform any unnecessary data, and analyse interesting patterns before performing Feature Engineering.

I examined the distribution of numerical and categorical features and found that the dataset contains **10 numerical features** and **11 categorical features**. I plotted various graphs to study their **distributions** and **statistical measures** (mean, standard deviation, minimum and maximum values, and percentiles for numerical features). I observed that the numerical features had different value ranges due to their distinct characteristics, and for each categorical feature, we had different values that were either objects or binary (yes/no). Next, I separately analysed numerical and categorical features to generate appropriate transformations for the project.

Numerical Features

- I plotted different **graphs**, including histograms with KDE plots, box plots, and standalone KDE plots for each numerical feature to visually understand their distribution, central tendency, spread, and density.
- I studied the **correlation** between features and found out that four variables had a high correlation (greater than ± 0.75). Since `emp.var.rate` was highly correlated with `cons.price.idx`, `euribor3m` and `nr.employed`, I choose to **keep `emp.var.rate`** due to its strong economic implications (it represents the quarterly variation in the employment rate in percentage), while **dropping the other three**.
- I explored the relationship between **numerical features** and the **target variable** `**y**` with histograms with density curves, box plots, and violin plots for each numerical feature in relation to “yes” and “no”. Additionally, I computed summary statistics of numerical features grouped by “yes” or “no”.

  For some numerical features, there were **significant differences in the distribution of “yes” and “no”** compared to the feature values (mean, standard deviation, and percentiles).

  I decided to **drop** the variable `duration` while keeping `previous` and `emp.var.rate`. This decision was made because `duration` represents the last contact duration with the client (in seconds) and is a measure that is not known before a call is performed. After the call ends, `y` is obviously known. Thus, this input should not be included in the dataset to maintain a realistic predictive model. Meanwhile, `previous` and `emp.var.rate` are significant measures, respectively representing the number of contacts before the marketing campaign and the employment variation rate, which can influence the client's final decision and thus my prediction.

Categorical Features

- I created **plots** (bar and pie charts) to understand the **distribution** of categorical variables and see if there were any unbalanced distributions. For example, the variable `default` was unbalanced with a 100% “no” and a 0% “yes” distribution of outcomes.
- I studied the **skewness** and the **mode proportions** for all the **categorical features**. Only the `default` feature had a skewness of more than 90% for one value. According to this, I **dropped `default`**.
- Then I explored the **relationship** between the **outcome** of each **categorical feature** and the **“yes” and “no” outcomes** of the target variable **`y`**. I found that some results could bias the target value. For example, the `poutcome` feature values “success” had a bias towards “yes”. This is compatible with the information we have regarding the variables; in fact, `poutcome` indicates the success or failure of past marketing campaigns with the client under consideration. Understandably, successful past campaigns will likely lead to more successful outcomes, while first-contact clients or those with previously failed campaigns will have fewer chances.

3. Data Pre-Processing and Feature Engineering

The data frame is almost ready to be used for Machine Learning algorithms, but the Exploratory Data Analysis showed that there is still some work to be done. Thus, Data Pre-Processing and Feature Engineering are the final modifications and transformations to the data before having the final data frame.

Two issues were still present in the data frame and needed to be resolved to apply Machine Learning methods effectively:

1) **Numerical features** had **very different scales of distribution** that couldn’t be simply compared.
   To solve this, I applied a **scaler (Standard Scaler)** to numerical features to bring each feature dimension into a comparable range with the others.
2) **Categorical features** had many **different outcome values** and **couldn’t be compared directly**.
   To address this, I performed **Feature Engineering** by converting categorical variables into numerical vectors that could be processed by Machine Learning algorithms. I used **One-Hot Encoding (Dummy Variables)**, a method that replaces categorical variables with features that have values of 0 or 1 for each category.

Now, the data frame had a new shape, with **more columns** due to the Dummy Variables: it had **30,488 rows and 43 columns**. There were **2 data types**: integers and floats. I **converted** everything **to float** to ensure maximum performance. At this point, all features were in the same format and ready to be compared. I calculated and plotted the **correlation heatmap** to identify any highly correlated features (greater than ±0.75) that I might have missed earlier. I found **three variables** with **significant correlation** measures, so I chose to **drop them** and re-compute the correlation.

In my view, the data frame was ready for the application of Machine Learning models: it contained only features with acceptable correlations, a unique data type (float), zero null values, values within a comparable range, and a sufficient number of entries for the purpose (30,488 rows and 40 columns).

4. Model Training and Evaluation

To apply Machine Learning algorithms to my data frame, I split it into training, test, and validation sets.
The data distribution was as follows:

- 80% of the total dataset as a training set and 20% as a test set;
- 25% of the training set as a validation set;

Final proportions out of the total data frame: **60% training set** | **20% test set** | **20% validation set**.
Before proceeding, I verified that the **distribution** of the target variable **`y` outcomes** was **similar** across all sets and consistent with the total data frame.

I decided to use four different Machine Learning models for the classification task:

- **Decision Tree Classifier**
  The **Decision Tree Classifier** works by splitting the data into subsets based on the value of input features, creating a tree-like model of decisions. It is ideal due to its interpretability, ability to handle diverse data types without assumptions about distribution, and capability to reveal feature importance, making it a solid baseline for understanding and predicting target variable `y` outcomes.

  After the first run, with an accuracy of 0.83 but 0.32 on the "yes" outcomes, I **fine-tuned parameters** such as **maximum depth** and **minimum samples per leaf** to avoid overfitting while maintaining performance.
  Qualitative analysis revealed that the model sometimes **misclassified** due to overfitting on the training data, particularly with less frequent outcomes. To mitigate this, I **pruned the tree** and **adjusted the depth**. Although **overall accuracy improved slightly**, the **F1 score for "yes" decreased**.

  The feature importance plot showed that the Decision Tree Classifier considered only a few (4) features.

- **Random Forest Classifier**
  The **Random Forest Classifier** is an ensemble learning method that builds multiple decision trees and merges their predictions. I **fine-tuned** the **number of trees** (n_estimators) and **max_features** to optimise performance.

  This model showed robust performance and reduced overfitting compared to a single decision tree. Errors were often due to noise or ambiguous data points. Error visualisations indicated misclassifications in cases where client behaviour patterns were less clear.

  Measures to improve performance included increasing the **number of trees** and **feature randomness**, resulting in high accuracy and stability. The **best performance on "yes" outcomes accuracy** was obtained after **redefining class weights** based on the **imbalance ratio ("no" count / "yes" count)**. However, this method decreased overall accuracy.
  Overall, I believe a higher F1 score for "yes" outcomes is preferable because the bank has better chances of finding more clients adhering to their campaign.

The **feature importance was more balanced**, with the model considering many more features compared to the single Decision Tree Classifier.

- **Logistic Regression**

  Logistic Regression models the probability of a binary outcome based on input features. It is well-suited for binary classification tasks due to its ability to output probabilities directly. It should perform well if the relationships between features and the target are linear but might underperform compared to ensemble methods on capturing complex, non-linear patterns.

  After the first performance, I got **high overall accuracy** but a **low F1 score on "yes" outcomes**. Thus, I used **class weights** to handle the class imbalance and got an improved F1 score for "yes".

  I applied **regularization (L2)** to prevent overfitting and used **cross-validation** to fine-tune the regularization parameter (C), aiming to improve the model's accuracy and performance on the validation set by identifying the best configuration.

  After hyperparameter tuning, the **performance metrics** for both the majority and minority classes **remained virtually unchanged** compared to the previous performance. The accuracy, recall, precision, and F1 scores didn't show significant improvement, suggesting that the hyperparameter tuning did not yield a better model configuration than before.

  This model provided good baseline performance and was less prone to overfitting but sometimes struggled with non-linear relationships in the data. Error analysis showed mispredictions mainly in cases with subtle patterns that linear models couldn't capture.

  The **feature importance** graph indicated that **Logistic Regression** gave importance to each feature, **considering the entire feature set**.

- **Neural Networks**

  Neural Networks, particularly a **Multi-Layer Perceptron (MLP)**, were used to capture complex patterns in the data. I experimented with **different architectures**, including varying the number of hidden layers and neurons, activation functions, and dropout rates to avoid overfitting. Despite being computationally intensive, Neural Networks excelled at capturing non-linear relationships from the first run of the model.

  Misclassifications were visualised to understand if they were due to insufficient training or overfitting. Measures to enhance performance included early stopping, learning rate adjustment, and data augmentation.

  The feature importance graph shows that **Neural Networks** models are the best for capturing complex patterns in the data and can **give importance to each feature**.

5. Conclusions

Decision Tree Classifier

```
Accuracy on validation set with best hyperparameters: 0.89

Classification Report:
              precision    recall  f1-score   support

         0.0       0.90      0.99      0.94      5344
         1.0       0.69      0.19      0.30       754

    accuracy                           0.89      6098
   macro avg       0.79      0.59      0.62      6098
weighted avg       0.87      0.89      0.86      6098
```

Random Forest Classifier

```
Accuracy on test set: 0.82

Classification Report on test set:
              precision    recall  f1-score   support

         0.0       0.94      0.85      0.89      5304
         1.0       0.38      0.63      0.47       794

    accuracy                           0.82      6098
   macro avg       0.66      0.74      0.68      6098
weighted avg       0.87      0.82      0.84      6098
```

Logistic Regression

```
Accuracy on validation set: 0.81

Classification Report:
              precision    recall  f1-score   support

         0.0       0.94      0.83      0.88      5344
         1.0       0.35      0.65      0.46       754

    accuracy                           0.81      6098
   macro avg       0.65      0.74      0.67      6098
weighted avg       0.87      0.81      0.83      6098
```

Neural Networks

```
Accuracy on validation set: 0.88

Classification Report:
              precision    recall  f1-score   support

         0.0       0.91      0.96      0.93      5344
         1.0       0.53      0.33      0.41       754

    accuracy                           0.88      6098
   macro avg       0.72      0.64      0.67      6098
weighted avg       0.86      0.88      0.87      6098
```

Each model was evaluated using performance metrics such as accuracy, precision, recall, and F1-score.

The **Random Forest Classifier** and **Logistic Regression** generally showed **better performance on F1-Score** for the **“yes” class** while the Decision Tree Classifier and Neural Networks showed higher general accuracy.

- The Random Forest achieved a balance between high precision and recall, especially for the “yes” class, with an overall accuracy of 0.82.
- Neural Networks demonstrated strong performance with an accuracy of 0.88, though struggled with recall for the minority class.
- Decision Trees, while interpretable, had lower recall for the “yes”.
- Logistic Regression offered a good baseline with an accuracy of 0.81 but similarly faced challenges with recall.

Ultimately, ensemble methods like Random Forest and complex models like Neural Networks provided superior performance in capturing intricate patterns within the data, while the **challenge** was **handling the imbalance between “yes” and “no” outcomes for the target variable `y`**. Logistic Regression and Neural Networks showed a more complete feature importance given to each feature, while Decision Tree and Random Forest Classifiers lacked in taking into account the importance of the whole feature set.