TERM DEPOSIT PREDICTION

Spring 2023

IIT Indore

END-EVALUATION REPORT

OVERVIEW

A bank's primary revenue stream comes from **Term Deposits**. A cash investment maintained by a financial institution is known as a Term Deposit. A predetermined period, or term, during which your money will be invested at an agreed-upon interest rate.

The bank uses a variety of outreach strategies, including email marketing, ads, telemarketing, and digital marketing, to reach out to its clients and sell them term deposits. The best of many ways to connect with consumers is through **telephone marketing**.

Nevertheless, a significant expenditure is necessary because enormous contact centers are used to carry out these operations.

Therefore, to target consumers with calls that are precisely directed at them, it is **essential to identify the customers who are most likely to convert in advance.**

DEFINITION

This problem involves building a binary classification model to predict the probability of a client subscribing to a term deposit. Given a set of features, such as client data and last contact information, etc., and a binary output feature, it is a supervised learning problem.

The goal is to minimize the calls by eliminating those who will not subscribe to the Term Deposit. This way, a company can save huge amounts of money and redirect to where necessary.

DATASET

The dataset used to predict Term Deposit subscriptions is related to direct telemarketing campaigns of a Portuguese Banking Institution, collected from 2008 to 2013. The Dataset consists of 16 predictors (9 categorical and 7 numeric) and 1 output. Out of these 16 predictors, 8 predictors are based on the Bank Client data (namely, **age, job, marital, education, default, housing, loan and balance**), 4 predictors are based on the last contact information (namely, **contact, month, day, duration**), 4 other attributes include **campaign, pdays, previous and outcome.**

There were copies of the same dataset present on the internet but with different numbers of data points. There was a balanced dataset with undersampled data and an imbalanced dataset present. We use the **imbalanced dataset** because it contains more data points.

REPOSITORY

GitHub Repository: https://github.com/mukulvain/Term-Deposit-Prediction

PREPROCESSING

Data Cleaning:

Fortunately, the dataset contains **no missing value** or any invalid entry**,** making it easier to preprocess the data.

Data Transformation:

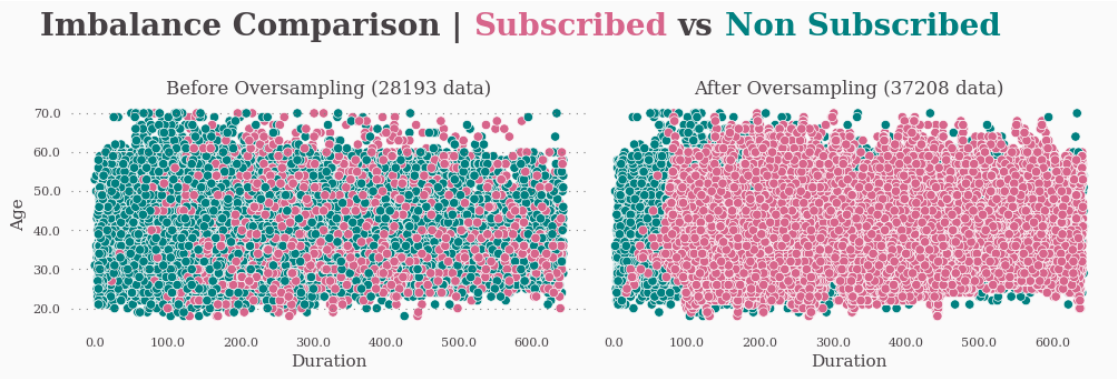
The categorical data is converted into a numerical format using **one-hot encoding**. One-Hot Encoding is used because there are not many categories within a feature. Thus, the dimensions and run-time would not be affected largely.

Normalization:

The numerical predictors are **scaled** to have zero mean and unit variance. Subsequently, the output variable is converted into 0 and 1.

Data Imbalance:

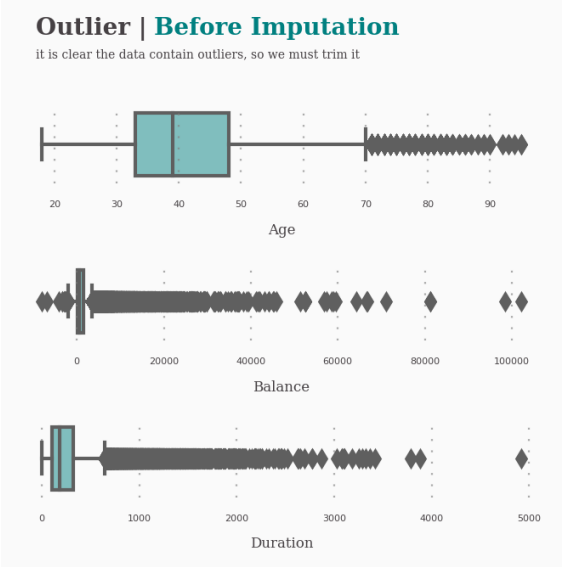
Since the dataset was imbalanced, we applied Oversampling using **SMOTE** to increase the data points. Undersampling was not used because undersampled data was already present on the Web.

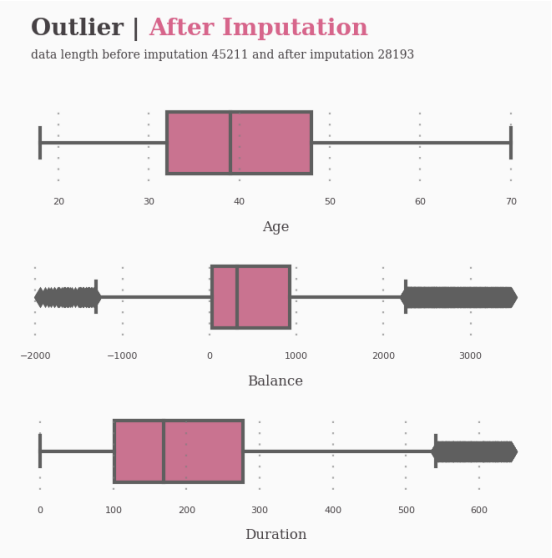


Data Splitting:

Stratified sampling is used to split the data into training and validation sets so that the percentage of an important feature in the training set is the same as in the validation set. This finalizes our dataset, which is then prepared to be worked upon.

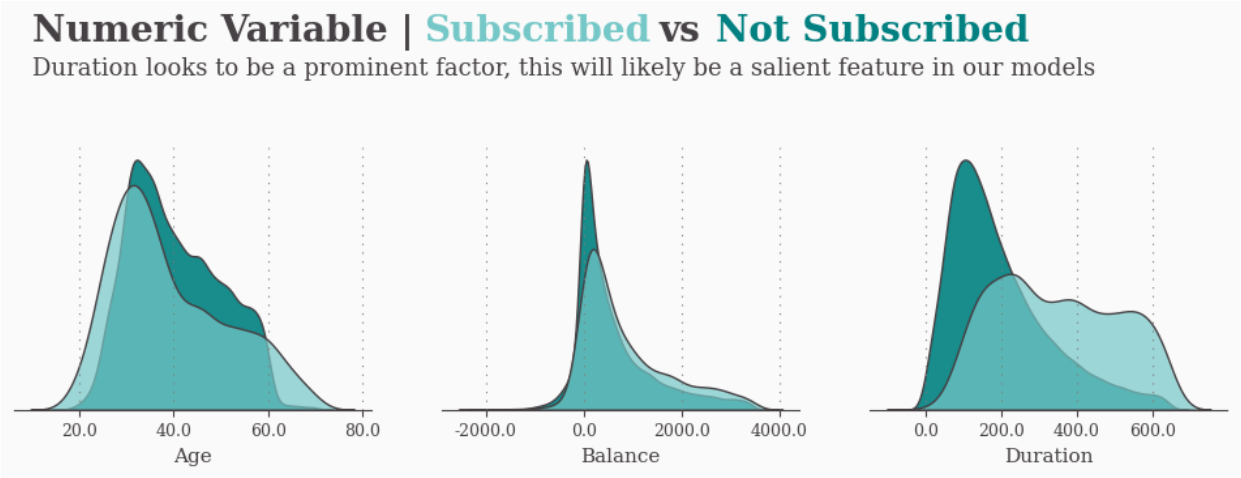
ANALYSIS



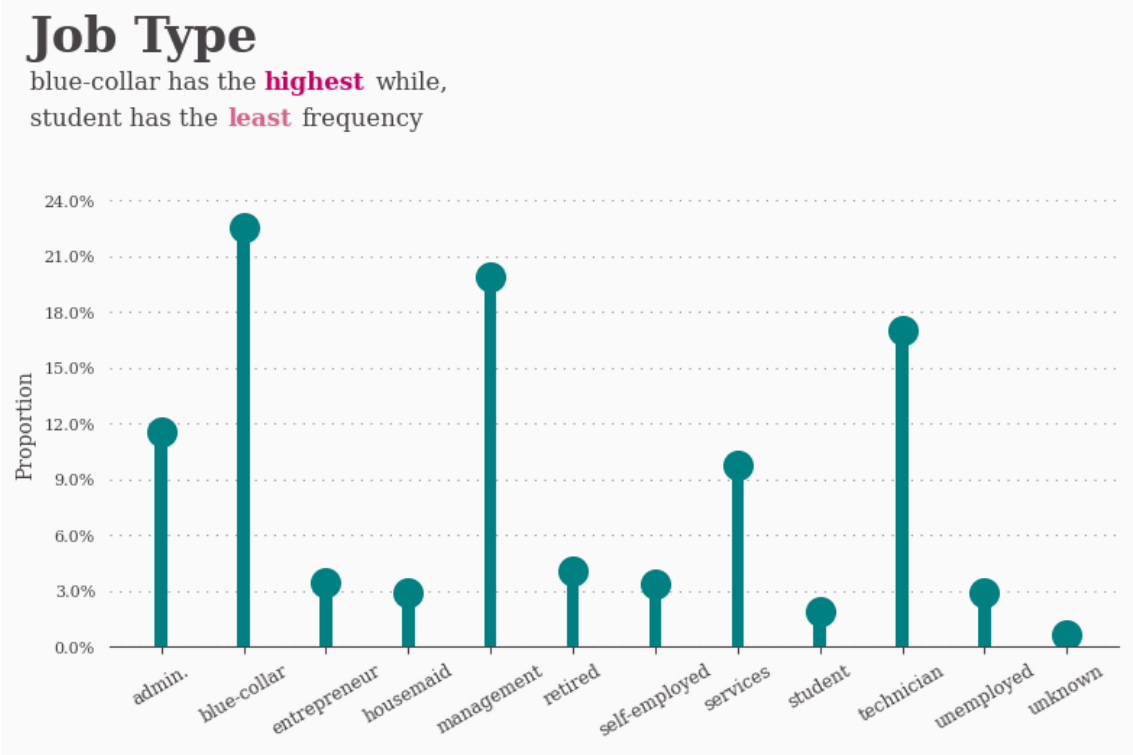


We found relations between Long term Deposits with Age, Balance and Duration while also being acknowledged the outliers present.

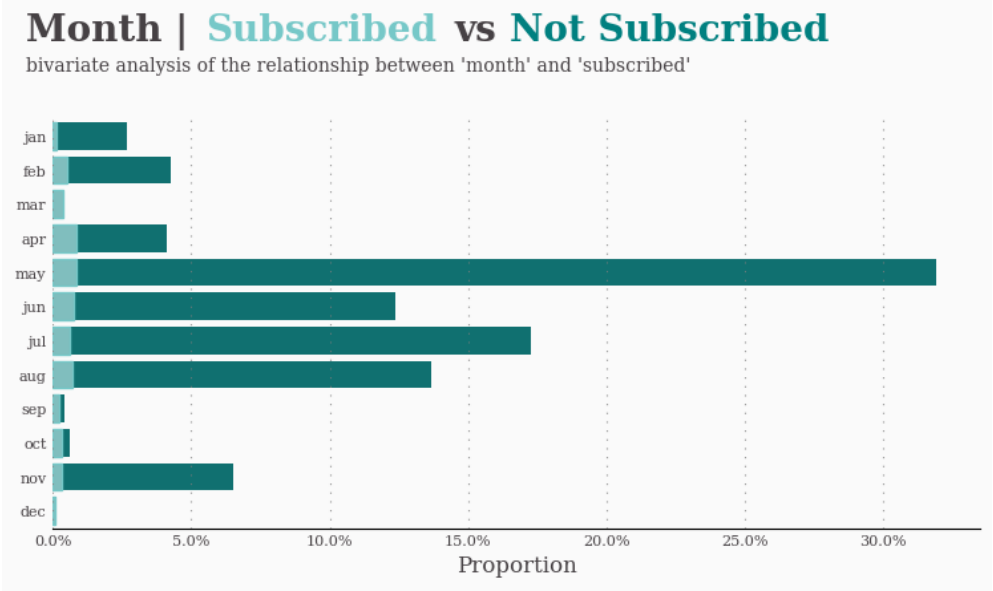
The Outliers were trimmed as a step to clean the data as outliers can lead to worse effects on the study, and trimming them refines the scope of the study.



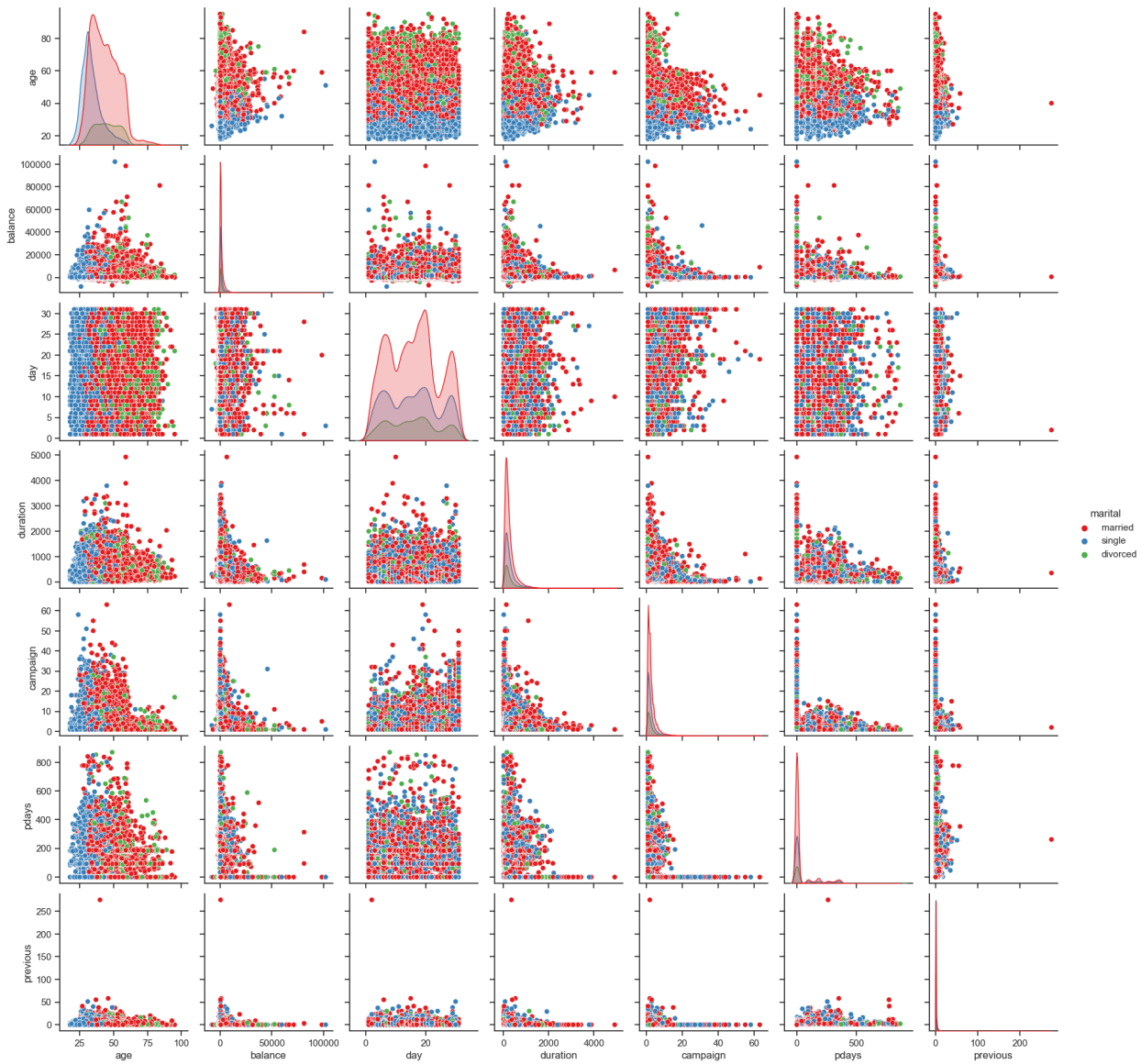
As seen in the graph above, the area of Age and Balance overlap more than Duration, it concludes that Duration can be a differentiating factor when it comes to Long term deposits



It is apparent from the above chart that middle-level type of employment plays a role in factoring if someone will subscribe or not.

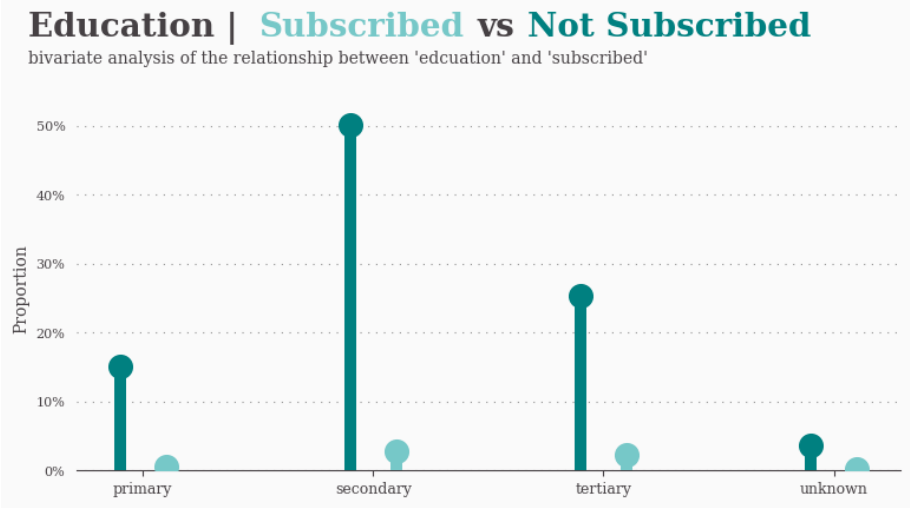


Doing a bivariate analysis of the deposits over months in a year, we can see how the annual nature of the fiscal cycle affects the probability of contacts and conversions

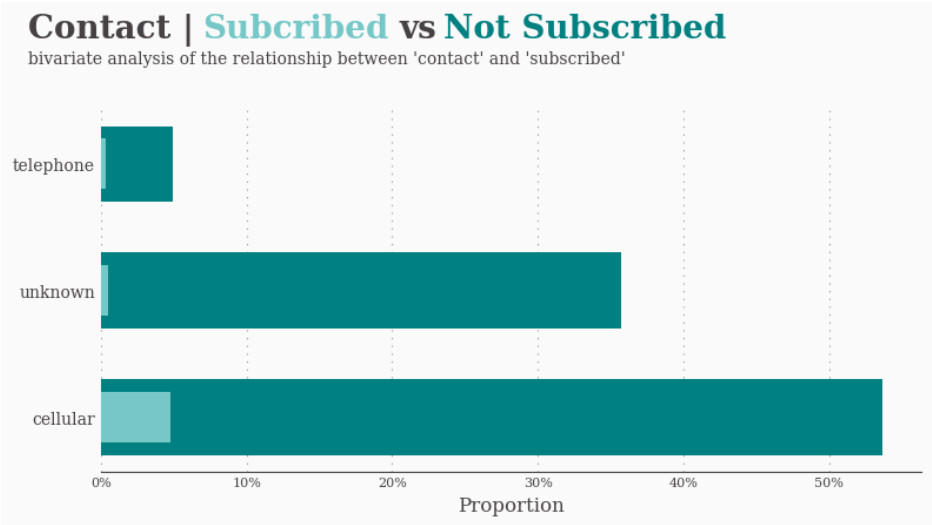


Plotting the graphs between marital status and all other attributes, we can see how it affects not only the subscription rate but also most of the other attributes.

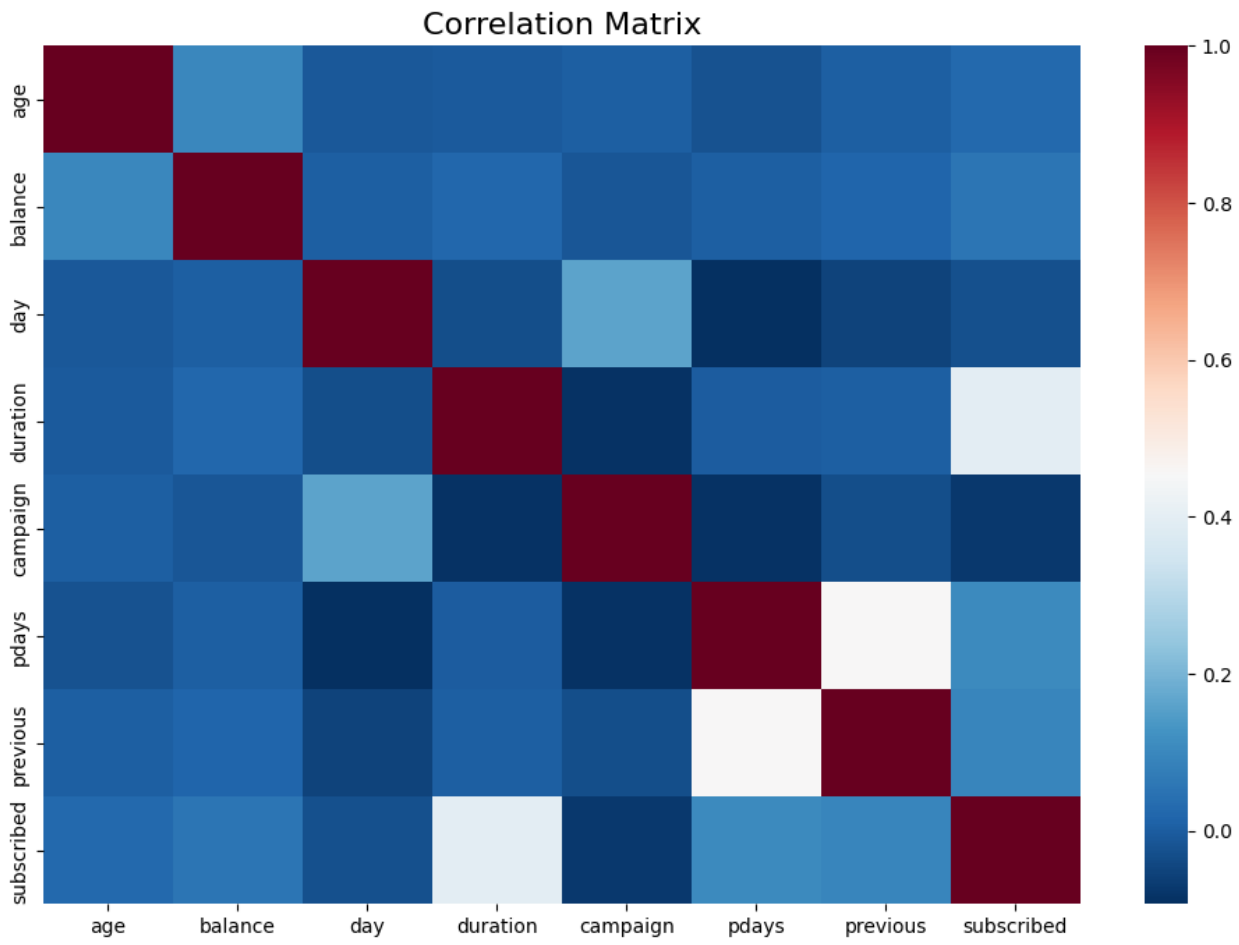
This signifies the effect of someone being single, married, or divorced over the finances of individuals or families and the psychological changes or implications on decisions made while it has a say on other factors. It indirectly induces its weightage on term deposits.



We looked at the variance of Long Term Subscriptions over the type of education spreading over tertiary, secondary, and primary.



Last but not least, means of contact don't imply much on conversions, but we gain insight into which modes are popular and thus will help banks to allocate resources efficiently.



There is no better way to conclude research results than a correlation matrix heatmap, which demonstrates how strongly attributes weigh on each other, going from colder to warmer colours and the diagonal being warmest as characteristics are compared to themselves.

- Here we can see age correlating with balance, which is obvious
- Also seen is the relation between day and campaign
- Duration and subscription are fundamentally related
- Lastly, pdays and previous attributes have about half a correlation

METRICS

Precision:

Precision is the ratio of true positives to the total number of positive predictions. In other words, it measures how many of the positive predictions made by the model were actually correct.

Recall:

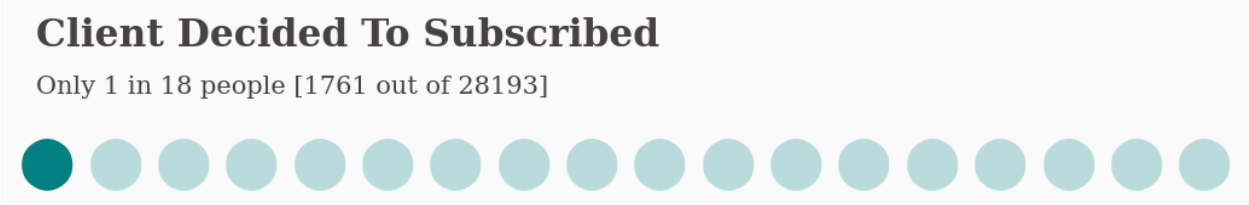
Recall is the ratio of true positives to the total number of actual positive cases. In other words, it measures how many of the positive cases the model was able to identify correctly.

Accuracy:

Accuracy is the ratio of correct predictions to the total number of predictions. In other words, it measures how many of the predictions made by the model were correct.

F1 Score:

F1 score is the harmonic mean of precision and recall. It combines Precision and Recall into a single metric that balances both measures.



As observed from this diagram, only **1 out of 18** people decide to make a Term Deposit.

We want to minimize the calls by eliminating those who will not subscribe to the Term Deposit. This way, a company can save vast amounts of money and redirect to where necessary.

Any company would not want to miss out on its potential customers. However, they would not hesitate to call a few customers who would not take the Term Subscription.

Thus, **we aim to reduce the False Negatives** using the **Recall** metric. However, we do not want to lose Precision after achieving a decent Recall score. Therefore, we wish to use a metric that incorporates Recall and the Precision metric.

F2 Score:

It places more emphasis on recall than precision, making it useful when the cost of false negatives (missed positive instances) is high.

The F2 score is calculated by taking the weighted harmonic mean of Precision and Recall, with a weight of 4 placed on Recall and 1 placed on Precision. This weight of 4 on Recall means that the F2 score is more sensitive to changes in Recall than Precision, making it a helpful metric when the goal is to maximize the detection of positive instances.

Mathematically, the F2 score can be expressed as

$$\textbf{F2 score} = (1 + \beta^2) * (\textbf{Precision} * \textbf{Recall}) / (\beta^2 * \textbf{Precision} + \textbf{Recall}),$$

Where β is a parameter that controls the weight given to recall compared to precision. In the F2 score, β is set to 2, which places more weight on recall.

ALGORITHMS

Linear Support Vector Machines:

SVM is a popular algorithm used for classification and regression problems. In this case, the goal is to predict whether a client will subscribe to a term deposit.

The Linear SVM algorithm finds the hyperplane that best divides the data into two classes. The hyperplane is chosen to maximize the margin between the two classes, where the margin is the distance between the hyperplane and the nearest data points from each class. The points that are closest to the hyperplane are called support vectors. The hyperplane is chosen in such a way that it maximizes the distance between the support vectors of the two classes.

If the data is not linearly separable, the SVM algorithm can find a hyperplane that best separates the data by introducing slack variables that allow some points to be misclassified. The **objective function** of the SVM algorithm is to minimize the sum of the slack variables while maximizing the margin between the two classes.

The Linear SVM algorithm is based on the use of a linear kernel function. This kernel function takes the dot product of two feature vectors as input and returns a scalar value. The dot product measures the similarity between two vectors. The linear kernel function is simple and efficient and can be used for datasets with many features.

Given a training dataset of n samples (x_1,y_1), (x_2,y_2), ..., (x_n,y_n), where x_i is a d-dimensional feature vector and yi is the binary class label (y_i = +1 or y_i = -1), the Linear SVM algorithm aims to find a hyperplane defined by $w^T x + b = 0$, where w is the weight vector, and b is the bias term, that best divides the data into two classes.

The optimization problem for the Linear SVM algorithm can be formulated as

$$\text{Minimize: } \|\mathbf{w}\|^2/2$$

$$\text{Subject to: } \mathbf{y_i(w^T xi + b) >= 1 \text{ for } i = 1,2,...,n}$$

where $\|w\|^2/2$ is the L2-norm of the weight vector w, and $y_i(w^T x_i + b)$ is the signed distance between the hyperplane and the i-th training sample.

We maximize the margin between the two classes by minimizing the L2-norm of the weight vector w. The constraints of the optimization problem ensure that all training samples are correctly classified and lie on the right side of the hyperplane.

The Linear SVM algorithm can be solved using various optimization techniques, such as quadratic programming or stochastic gradient descent.

Once the optimization problem is solved, the weight vector w and bias term b can be used to predict the class label of new samples using the equation $y = \text{sign}(w^T x + b)$, where sign() is the sign function that returns +1 or -1 depending on the sign of its argument.

Naive Bayes:

Naive Bayes is a probabilistic classification algorithm that uses Bayes' theorem to calculate the probability of a target class based on the features of the input data. In the context of the given data on direct marketing campaigns of a Portuguese banking institution, the Naive Bayes algorithm can be used to predict whether a customer will subscribe to a term deposit or not.

The Naive Bayes algorithm assumes that the features are conditionally independent given the target class, meaning that the presence of one feature does not affect the presence of other features. This assumption simplifies the calculation of the conditional probabilities, making the algorithm computationally efficient and scalable to large datasets.

The Naive Bayes algorithm is a probabilistic algorithm used for classification. It is based on Bayes' theorem, which is:

$$P(C|X) = P(X|C) * P(C) / P(X)$$

where C is the class, X is the feature vector, $P(C|X)$ is the posterior probability of class given the feature vector, $P(X|C)$ is the likelihood of the feature vector given the class, P(C) is the prior probability of the class, and P(X) is the evidence probability.

The Naive Bayes algorithm makes a naive assumption that all the features are independent of each other given the class label. Hence, the likelihood of the feature vector given the class can be calculated as

$$P(X \mid C) = P(x1 \mid C) * P(x2 \mid C) * \ldots * P(xn \mid C)$$

where x_i is the ith feature of the feature vector.

Using this assumption, we can rewrite Bayes' theorem as

$$P(C \mid X) = P(x1 \mid C) * P(x2 \mid C) * \ldots * P(xn \mid C) * P(C) / P(X)$$

To classify a new instance, the Naive Bayes algorithm calculates the posterior probability of the instance belonging to each class. It selects the class with the highest probability.

The algorithm estimates the prior probability of each class from the training data and the likelihood of each feature given to each class. It then uses these probabilities to classify new instances.

XGBoosting over Grid Search CV:

Gradient boosting is a machine learning technique that combines multiple weak models (often decision trees) to create a more accurate and robust model. It works by iteratively adding models to the ensemble, with each new model focusing on the errors of the previous model. This is done by calculating the gradient of the loss function with respect to the predictions of the previous model, and then fitting a new model to the residuals. The final prediction is then the sum of the predictions from all the models in the ensemble.

The formula for gradient boosting can be represented as:

$$F(x) = \Sigma\ f\_i(x)$$

where $F(x)$ is the final prediction, Σ represents the sum over all the weak models $f\_i(x)$, and x is the input features. At each iteration, the model fits a new weak model $h(x)$ to the negative gradient of the loss function $L(y, F(x))$, where y is the target variable. The negative gradient represents the direction in which the model needs to improve. The weak model $h(x)$ is then added to the ensemble and scaled by a learning rate α. The final prediction $F(x)$ is the sum of all the weak models in the ensemble.

Grid search CV (Cross Validation) is a hyperparameter tuning technique in machine learning that exhaustively searches through a specified hyperparameter grid to find the optimal combination of hyperparameters for a given model. It involves defining a grid of

hyperparameters and evaluating the model's performance with each combination of hyperparameters using cross-validation. Cross-validation involves splitting the data into several subsets and training the model on a combination of these subsets. The process is repeated multiple times, and the average performance is used to evaluate the model. Grid search CV is a powerful and widely used technique to improve the performance of machine learning models.

The method for grid search CV can be represented as:

```
for each hyperparameter combination in the hyperparameter grid:
   for each fold in the cross-validation:
       fit the model with the hyperparameter combination on the training set
       evaluate the model on the validation set
   calculate the average performance across all folds
select the hyperparameter combination with the highest average performance
```

XGBoost is a powerful machine learning algorithm that is used for regression and classification problems. It stands for "Extreme Gradient Boosting" and is a variant of the gradient boosting algorithm. XGBoost uses a regularized gradient boosting framework, which makes it resistant to overfitting and enhances its ability to generalize to new data. It works by iteratively adding weak models (often decision trees) to an ensemble and updating the model with the gradients of the loss function, making it more accurate with each iteration. XGBoost is known for its speed and scalability, and is widely used in industry and academia for various machine learning tasks.

XGBoost is a machine learning algorithm that is known for its speed and performance, and it is often used in combination with grid search CV to optimize its hyperparameters. Grid search CV is a hyperparameter tuning technique that involves searching through a grid of hyperparameters to find the optimal combination for a given model. XGBoost has several hyperparameters that can be tuned to improve its performance, such as learning rate, maximum depth, and regularization parameters. Grid search CV can be used to find the best combination of these hyperparameters for XGBoost, which can lead to improved

accuracy and better generalization to new data. By combining the power of XGBoost with the optimization capabilities of grid search CV, we can create highly accurate and robust machine learning models.

The formula for XGBoost can be represented as:

```
F_t(x) = F_{t-1}(x) + α_t * h_t(x)

where

   F_t(x) is the predicted target value at time t

   F_{t-1}(x) is the predicted target value at time t-1

   α_t is the learning rate at time t

   h_t(x) is the weak model at time t
```

Random Forest:

Random Forest is an ensemble learning method that combines multiple decision trees to make a final prediction. Let's assume we have a dataset with `N` samples and `M` features, where each sample `i` is represented by a feature vector `x_i = (x_{i1}, x_{i2}, ..., x_{iM})`, and the corresponding label is `y_i`. The goal of the Random Forest algorithm is to learn a function `F(x)` that can predict the label `y` for a new input feature vector `x`.

Random Forest builds `T` decision trees, where each tree is trained on a different bootstrap sample of the original dataset. A bootstrap sample is created by randomly sampling `N` samples from the original dataset with replacement, meaning that some samples may be included multiple times in the sample, while others may be excluded.

At each node of the tree, a subset of `m` features is randomly selected from the `M` features, and the best feature and split point is chosen based on a splitting criterion such as the Gini index or information gain. The number `m` is typically much smaller than `M`, which helps to reduce the correlation between the decision trees and improve the model's generalization ability.

The prediction of a new input feature vector `x` is made by aggregating the predictions of all `T` decision trees. For classification tasks, the final output is obtained by majority vote, i.e., the class that is predicted by the most number of trees. For regression tasks, the final output is obtained by averaging the predictions of all trees.

The Random Forest algorithm can be summarized mathematically as follows:

1. For each tree `t` in `T`:
    - Sample a bootstrap dataset `D_t` from the original dataset `D` of size `N`.
    - Randomly select a subset of `m` features from `M` features.
    - Train a decision tree `h_t(x)` on the bootstrap dataset `D_t` using the selected features.
2. The prediction for a new input feature vector `x` is obtained by aggregating the predictions of all trees:
    - For classification tasks: `F(x) = mode(h_t(x)) for t in T`.
    - For regression tasks: `F(x) = 1/T * Σ h_t(x) for t in T`.

Where `mode` represents the most frequent class predicted by the trees, and `Σ` represents the sum over all trees. Random Forest is a versatile and effective machine learning algorithm that is widely used in various applications due to its high accuracy and robustness.

RESULTS

Method	Accuracy (IN %)	F2 Score
Naive Bayes	70.34	0.37
Logistic Regression	81.19	0.41
Decision Tree	82.85	0.31
XGBoost	88.50	0.41
Random Forest	97.15	0.92

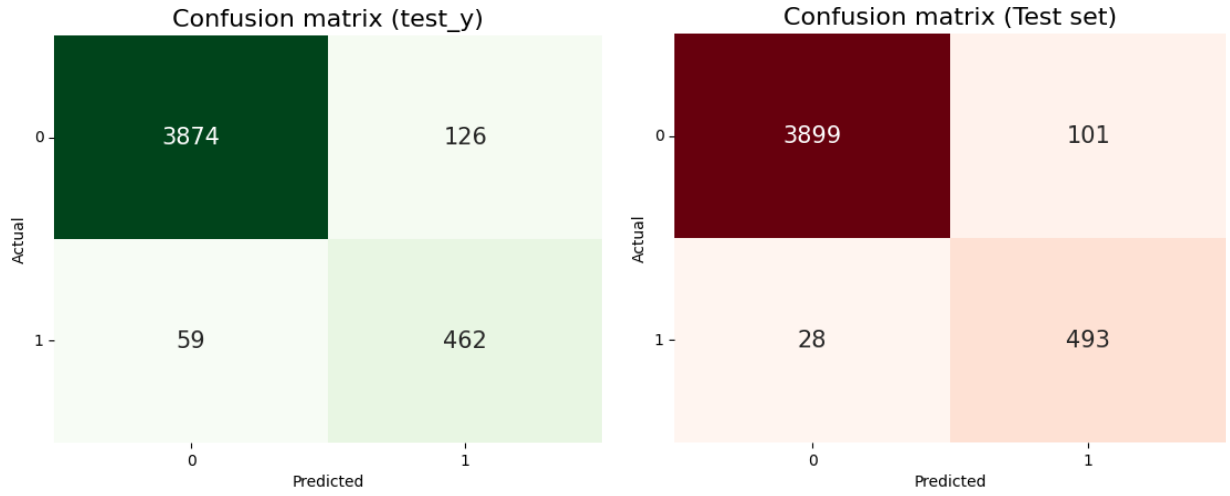
Clearly, the results obtained by Naive Bayes, Logistic Regression and Decision are not at par.

From a research point of view, the accuracy had to be increased. We were advised to use Ensemble Methods or Boosting Methods that would bring about better results.

XGBoost brought about a good improvement in the percentage points of accuracy. However, it was still insufficient, and the accuracy was under 90%.

On the other hand, Random Forest gave surprisingly superior results with great accuracy and even better than some of the Research papers.

Confusion Matrix:



Comparison:

S. No.	Classification Models	Evolution Metrics of Classification Models				
		Accuracy	Precision	Recall	F1-Score	AUROC -score
1.	LR	92.72	70.58	54.54	61.53	93.62
2.	GNB	85.0	38.55	72.72	50.39	84.76
3.	RF	91	59.37	43.18	50.0	93.53
4.	SVM	90.53	58.62	38.63	46.57	91.4
5.	DT	91.0	56.75	47.72	51.85	71.69

Source [4]

Table 1. Comparison result of Neural Network, SVM, Naïve Bayes and Stacking with Ensemble methods

		Accuracy (%)	Error-rate (%)	Sensitivity (%)	Specificity (%)
Neural Network	NN	94.8684	5.1316	95.5175	98.225
	NN (Bagging)	96.6158	3.3842	97.14	99.075
	NN (Boosting)	94.8684	5.1316	94.21	98.275
SVM	SVM	89.7589	10.2411	85.20	96.875
	SVM (Bagging)	89.8031	10.1969	89.84	96.95
	SVM (Boosting)	90.1198	9.8872	90.50	97.00
Naive Bayes	NB	88.2327	11.7673	84.32	94.2
	NB (Bagging)	88.3654	11.6346	87.40	94.325
	NB (Boosting)	88.7193	11.2807	89.70	95.025
Stacking	Classifier (SVM, NN) Meta classifier (SVM)	91.3294	8.6706	89.45	98.975

Source[1]

In comparison to these papers, our results achieved using Random Forest produced better results in terms of accuracy.

FURTHER SCOPE

Feature Engineering:

A critical aspect of building a robust predictive model is selecting the right features that strongly correlate with the target variable. In this case, we can explore different data sources and incorporate additional features to help us better predict the likelihood of a client subscribing to a term deposit. For example, we can extract information from the client's social media profiles, web browsing history, or transaction data to build a more comprehensive customer profile.

Deployment and Integration:

Once we have developed a robust predictive model, the next step is to deploy it in a real-world setting and integrate it with the bank's existing systems. We can use APIs and microservices to expose the model as a web service and integrate it with the bank's CRM or call center software. We can also develop a user-friendly interface that allows business users to query the model and generate reports on model performance and predictions.

Ethical Considerations:

It is essential to consider the ethical implications of using predictive models to target customers. For example, we must ensure that the model does not discriminate against any particular customer group based on age, gender, ethnicity, or other protected characteristics. We also need to ensure that the model's predictions are transparent and explainable so that customers can understand how their data is used to make decisions about them.

Similar Problems:

Similar Models can be developed for

1. Credit Risk Assessment
2. Churn Prediction
3. Fraud Detection
4. Customer Segmentation
5. Inventory Management

REFERENCES

1. Patwary, M. J., Akter, S., Alam, M. B., & Karim, A. R. (2021). Bank Deposit Prediction Using Ensemble Learning. Artificial Intelligence Evolution, 42-51.
2. Muslim, M. A., Dasril, Y., Alamsyah, A., & Mustaqim, T. (2021, June). Bank predictions for prospective long-term deposit investors using machine learning LightGBM and SMOTE. In Journal of Physics: Conference Series (Vol. 1918, No. 4, p. 042143). IOP Publishing.
3. Term Deposit Subscription Prediction | Thean C. Lim.
4. Borugadda, P., Nandru, P., & Madhavaiah, C. (2021). Predicting the success of bank telemarketing for selling long-term deposits: An application of machine learning algorithms. *St. Theresa Journal of Humanities and Social Sciences*, *7*(1), 91-108.

TEAM

1. Mukul Jain (200001050)
2. Nilay Ganvit (200001053)

Under the guidance of Dr. Aruna Tiwari, Professor, Computer Science and Engineering, IIT Indore.