**Title:** Term deposit lead prediction

**Client Information:** The investment and portfolio department of the Bank of Portugal (Actual) (*Change name by keeping same format*)

**Problem Statement:** The investment and portfolio department of a bank would want to identify their customers who potentially would subscribe to their term deposits.

This activity aims to find a machine learning model that can predict which future clients would subscribe to their term deposit. This will increase the bank's campaign efficiency and effectiveness.

The bank would identify customers who would subscribe to their term deposit and use that knowledge to direct their marketing efforts to them. This is known as target marketing.

This would help them better manage their resources and avoid waste.

**What is a term deposit?**

With a term deposit, you lock away an amount of money for an agreed length of time (the 'term') – that means you can't access the money until the term is up. In return, you'll get a guaranteed rate of interest for the term you select, so you'll know exactly what the return on your money will be.

**Data Collection:**

**EDA:**

Exploratory Data Analysis (EDA) is an approach to analyse the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

Step 1: First, we will import all the python libraries that are required for this, which include NumPy for numerical calculations and scientific computing, Pandas for handling data, and **Matplotlib** and **Seaborn** for visualization.

Step 2: Then we will load the data into the Pandas data frame.

Step 3: We can observe the dataset by checking a few of the rows using the head() method, which returns the first five records from the dataset.

Step 4: Using .shape, we can observe the dimensions of the data.

Step 5: .info() method shows some of the characteristics of the data such as Column Name, No. of non-null values of our columns, Dtype of the data, and Memory Usage.

Step 6: We will use describe() method, which shows basic statistical characteristics of each numerical feature (int64 and float64 types): number of non-missing values, mean, standard deviation, range, median, 0.25, 0.50, 0.75 quartiles.

Step 7: Missing value visualisation: Missing no library

Step 8: Nature of distribution visualisation: Histogram, sns.distplot()

**Step 9: Outlier Visualisation: Box plot**

Step 10: Dataset Imbalance Visualisation: Countplot

**Step 11: Correlation Visualization: sns.heatmap()**

**Step 12: Hypothesis testing**

**Step 13: Kde Plots**

## Feature Engineering

In Data Science, the performance of the model is depending on data pre-processing and data handling. Suppose if we build a model without Handling data, we got an accuracy of around 70%. By applying the Feature engineering on the same model there is a chance to increase the performance from 70% to more.

Steps:

1. Feature Selection
2. Handling missing values
3. Handling imbalanced data
4. Handling outliers
5. Binning
6. Encoding
7. Feature Scaling

**1. Feature selection:** Feature selection is nothing but a selection of required independent features. Selecting the important independent features which have more relation with the dependent feature will help to build a good model.

Benefits of Feature selection:

1. It helps in avoiding curse of dimensionality

2. It helps to reduce overfitting

3. It reduces training and testing

4. It improves accuracy

## TSNE

T-distributed Stochastic Neighbor Embedding (TSNE) reduces the reduced dimensions based on non-linear local relationships among the data points.

## Autoencoders

Autoencoder is an unsupervised artificial neural network trained with the back-propagation algorithm to reproduce the input vector onto the output layer. Its procedure starts compressing the original data into a short-code, ignoring noise using an encoder. This is followed by an algorithm that decompresses the short-code to generate the data as close as possible to the original input using the decoder.

## PCA

Principal Component Analysis (PCA) is a statistical process that perpendicularly transforms the original numeric columns of data into a new set of principal components. It is a linear dimensionality reduction technique.

The dimensionality reduction technique chosen is **Autoencoders**. Autoencoder output can compress the information better into low dimensional latent space, leveraging its ability to model complex nonlinear functions. The resulting dataset from autoencoders gives higher accuracy than the other techniques.

With t-distributed stochastic neighbour embedding, the dimensions a fewer for a dataset that originally had 21 features. This is seen with its reduction in accuracy for the machine learning models used. It has the lowest accuracy of the three techniques.

With principal component analysis, the model's accuracy is lower than that with the autoencoder when applying the resulting dataset from the principal component analysis.

## 2.Handling Missing Values

Many machine learning algorithms fail if the dataset contains missing values. However, algorithms like K-nearest and Naive Bayes support data with missing values. You may end up building a biased machine learning model which will lead to incorrect results if the missing values are not handled properly. Therefore, it is important to handle missing values in a proper manner.

We have used **Iterative** imputer for Missing value handling, other methods are listed below.

1. Deleting missing values
2. With help of Pandas: Mean, Mode, Median
3. Imputers: Simple, Iterative. KNN
4. Datawig Liabrary
5. Using Feature Engine

**3. Handling imbalanced dataset:**

Balancing a dataset makes training a model easier because it helps prevent the model from becoming biased towards one class. In other words, the model will no longer favour the majority class just because it contains more data.

In our case we are doing multiclass classification, in which there is one predominant class. So, we have to handle it with **SMOTE**.

**SMOTE** and other Multiclass classification's dataset imbalance handling methods are provided in below link:

https://machinelearningmastery.com/multi-class-imbalanced-classification/

**4. Handling outliers USING PyOD TO CLEAN OUTLIERS**

Outliers in data may contain valuable information. Or be meaningless aberrations caused by measurement and recording errors. In any case, they can cause problems, so it's important to question and analyse outliers.

How to handle?

- Trimming/removing the outlier
- Quantile based flooring and capping
- Mean/Median imputation

A common practice of detecting outliers is using box plots, histograms, and scatter plots. This methodology may not be efficient in large datasets. PyOD, a python library has different algorithms that can be used to detect and clean outliers.

**5. Discretisation/ Binning**: Binning or discretization is used to transform a continuous or numerical variable into a categorical feature. Binning of continuous variables introduces non-linearity and tends to improve the performance of the model. It can also be used to identify missing values or outliers.

Methods:

1. Fixed width binning
2. Quantile binning
3. Binning by instinct
4. K-Means Binning
5. Decision Tree Binning

**6. Data Transformations:** Data transformation is the process of converting raw data into a format or structure that would be more suitable for model building and also data discovery in general.

Why need data transformation?

- The algorithm is more likely to be biased when the data distribution is skewed
- Transforming data into the same scale allows the algorithm to compare the relative relationship between data points better.

Types:

1. Log Transformation
2. Exponential Transformation
3. Square-Root Transformation
4. Reciprocal Transformation
5. Box-Cox Transformation
6. Yeo-Johnson Transformation

https://www.analyticsvidhya.com/blog/2021/05/feature-transformations-in-data-science-a-detailed-walkthrough/

**7. Feature Encoding:** Machine learning models can only work with numerical values. For this reason, it is necessary to transform the categorical values of the relevant features into numerical ones. This process is called feature encoding.

Types:

1. Ordinal: Label encoding
2. Nominal: One hot encoding; Dummy encoding; Target Encoding.

We have done label encoding for all categorical features except marital status & personal_loan. Since, both are yes/no.

For marital status & personal_loan we have done custom mapping.

**8. Feature Scaling:** Feature scaling is a method used to normalize the range of independent variables or features of data. Following are the benefits of FS.

- Scaling of the data makes it easy for a model to learn and understand the problem.
- Feature Scaling is done to bring the features in the dataset into a finite range.
- Scaling the features makes the flow of gradient descent smooth and helps algorithms quickly reach the minima of the cost function.
- Feature scaling can significantly boost a model's performance.

Types:

1. Absolute Maximum Scaling
2. Min-Max Scaling
3. Normalization
4. Standardization
5. Robust Scaling **6. Quantile transformer**

**Train Test Split:** 80-20 Split with stratification of Target

## Classification Model

• Algorithm Used:

Xgboost, CatBoost, lightGBM

Hyperparameter tuning with **optuna**

https://www.analyticsvidhya.com/blog/2020/11/hyperparameter-tuning-using-optuna/

Feature selection and Hyper parameter tuning are the key aspects to build a robust Machine Learning Model. Firstly, we get the feature importance of each feature of the dataset by using the feature importance property of the model. Feature importance gives a score for each feature of the data, the higher the score more important or relevant is the feature towards the output variable.

• Correlation states how the features are related to each other or the target variable. Correlation between remaining features is visualised through Heatmap. Among the two features having high correlation, we eliminated one of it and kept the other.

• Outliers are visualised using Box Plot and then removed as they were generating high skewness in the data.

In all the models selected even with hyperparameter tuning, (any of your choice) Classifier had higher accuracy scores. The figure below shows the different metric scores of the (any of your choice) model used. There was a slightly significant difference in the scores while using Stratified k-fold and k-fold cross-validation. Stratified k-fold had higher scores compared to k-fold.

```
Accuracy scores of the model: 0.88

 Classification report of the model
-------------------------------------
              precision    recall  f1-score   support

           0       0.89      0.87      0.88      3442
           1       0.88      0.90      0.89      3537

    accuracy                           0.88      6979
   macro avg       0.89      0.88      0.88      6979
weighted avg       0.89      0.88      0.88      6979


 Confusion Matrix of the model
-------------------------------------
[[2993  449]
 [ 354 3183]]
```

VCS: Git

Framework - Django

Cloud Platform – Heroku