Dublin Business School

PREDICTION FOR TERM DEPOSIT SUBSCRIPTION– PROJECT REPORT

WRITTEN BY,
MAYAKRISHNAN PERUMAL .

PREDICTION FOR TERM DEPOSIT SUBSCRIPTION - PROJECT REPORT

Name: Mayakrishnan Perumal

Student Number: 20055470

Class: Group C (2025 - 2026)

Programme: MSc in Data Analytics

Module: Machine Learning and Pattern Recognition

LIST OF CONTENTS

CHAPTER - I

INTRODUCTION

This project aims to apply python programming skills for data analysis. The main objective is to find the future output for random inputs using Machine Learning models, Hyperparameter tuning and overfitting avoidance mechanisms. The dataset used in this prediction was sourced from Kaggle.

1.1. PURPOSE OF THIS PROJECT

The purpose of this project is to predict the future output using the inputs in the dataset. Using Machine Learning models, we aim to predict whether the customers subscribe the term deposit scheme or not. The project may help the bank sectors know the customer who will subscribe the term deposit. Using this prediction, the bank sector will increase their term deposit subscribers.

1.2. OBJECTIVES OF THE ANALYSIS

The core objectives of this project:

a) **Data preparing**
Prepare the data to apply the machine learning models using data encoding, cleaning, scaling and splitting and prepare the new data to find the output using Machine Learning tools.

b) **Apply Machine Learning Models**
Find the accuracy of our prepared data using Machine Learning tools such as Logistic Regression, Random Forest Classifier, Decision Tree Classifier and Support Vector Classifier.

c) **Overfitting avoidance**
Avoid the overfitting problems using cross validation and Grid searchCV. It will helps to get the better accuracy and best output for our random inputs.

d) **Predict the output for new data**
Predict the output using the Machine Learning Models after applying overfitting avoidance mechanisms and finding the best model for new input data.

CHAPTER - II

DATA DESCRIPTION

2.1. SOURCE OF THE DATASET

The dataset used in this project is derived from a real-world Portuguese retail bank's direct marketing campaigns(telephone calls) between May 2008 and November 2010 in Portugal. This dataset was originally available on Kaggle under the title "Bank Marketing Dataset ". This dataset was first published in the **UCI Machine Learning Repository** by **Sérgio Moro, Paulo Cortez, and Paulo Rita** in their research paper under the title *"A Data-Driven Approach to Predict the Success of Bank Telemarketing"* (2014). The bank staff was called to the all customers of the bank and confirmed that whether the customer subscribed the term deposit(fixed deposit account) or not.

2.2. DATASET DESCRIPTION

The dataset is structured tabular data and each row representing a single customer's data entry. It contains **4521 rows** and **17 features**, all of which are clearly explained and prepared for analysis. The dataset includes both numerical and categorical data. The **y** column is the target variable for classification type and it refers the term deposit subscription.

2.3. EXPLANATION OF EACH FEATURE

SI.NO	Feature name	Data type	Explanation
1	Age	integer	Customer's age
2	Job	object	Types of customer's job
3	Marital	object	Marital status
4	Education	object	Customer's education level
5	Default	object	A binary value: yes means customer has credited in default, no means not credited.
6	Balance	integer	Average yearly balance
7	Housing	object	A binary value: yes means customer had housing loan, no means not get the loan.
8	Loan	object	A binary value: yes means customer had personal loan, no means not get the loan.
9	Contact	object	Communication type between bank and customer.
10	Day	integer	Last contact day of the month(1- 31)
11	Month	object	Last contact of the year ("Jan", "Feb",….,"Dec")
12	Duration	integer	Last contact duration in seconds
13	Campaign	integer	Number of contacts performed before this campaign.
14	Pdays	integer	Number of days passed after the customer was last contacted (-1 means customer was never contacted)

15	Previous	integer	Number of contacts before this campaign.
16	Poutcome	object	Outcome of the previous marketing campaign.
17	Y	object	A binary value: yes - customer was subscribed the term deposit, No- not subscribed.

CHAPTER - III

DATA ANALYSIS

3.1. DATA PREPARATION

3.1.1. DATA READING

First of all, we need to upload the dataset in the platform(Colab). Then we read the dataset using ***read_csv*** function from ***pandas*** library. After reading the file, we can find the information like null values, counting of rows and columns, data types of all columns using ***.info()*** function. If we have any error data like null values or unrealistic values like negative values in the age column, we will need to clean the data. Fortunately, we haven't any error data in our dataset.

Information of our dataset by using code:

```
Range Index: 4521 entries, 0 to 4520
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
0   age         4521 non-null   int64
1   job         4521 non-null   object
2   marital     4521 non-null   object
3   education   4521 non-null   object
4   default     4521 non-null   object
5   balance     4521 non-null   int64
6   housing     4521 non-null   object
7   loan        4521 non-null   object
8   contact     4521 non-null   object
9   day         4521 non-null   int64
10  month       4521 non-null   object
11  duration    4521 non-null   int64
12  campaign    4521 non-null   int64
13  pdays       4521 non-null   int64
14  previous    4521 non-null   int64
15  poutcome    4521 non-null   object
16  y           4521 non-null   object

    Data types: int64(7), object(10)
```

3.1.2. DATA ENCODING

Machine learning tools are work only for data which type is numerical (int, float). In our dataset, we have 7 integer type and 10 object type columns, So we need to change the 10 object type columns to numerical type.

In these 10 columns, four columns were classification type (which column have only two different values like '**Yes**' and '**No**'). We need to change that values to numbers like "**0**" and "**1**" using ***.map{}*** function to change the data type from object to integer type for use machine learning tools. If we have any column which values are looks like gradings, we can change the values to 1 to n using ***.map{}*** function. Here we have one column which name is **Education** that have 4 different values like gradings (The values: unknown~1, primary~2, secondary~3, teritary~4). We put the suitable number for the values instead of string data(Alphabetic) and this could be help to change the object type column to integer type. Now we found the solution to change the data type of 5 columns out of 10 object columns to integer type.

In the remaining five columns, we have three columns which have below than 10 different values. In this case, we can use ***get_dummies*** function from ***pandas*** library for change the data type of these columns to integer type. If we use this function, it will make a new columns for each different values. After that, we can drop the first column of the new columns using ***drop_first= True*** function to get a good accuracy while using machine learning tools. Now we found the solution to change the data type of 8 columns out of 10 object columns to integer type.

In the remaining two columns have more than 10 different values. If we use the ***get_dummies*** function for this two columns, it will be make more new columns and it is not help to get a best accuracy. We need to use the ***TargetEncoder*** function by install the ***category_encoders*** library for avoid the under fitting problems. This function gives the common values in numeric according to the target variable. If we want to use ***TargetEncoder*** function, we can divide the data first.

3.1.3. DATA DIVIDING

We need to find and separate the input and output variables of our dataset in our project. In this dataset output (Target) variable is "***y***" and other variables are input variables. We could divide the data to x as input and y as output by using the ***.drop()*** function.

3.1.4. DATA SCALING

Data scaling is an important tool for Machine Learning. Most of the dataset have non-familiar data, so we need to scale the data for make the data are familiar to each other. In this case, we used Z score standardization formula by using ***StandardScaler*** function from *preprocessing* in ***sklearn*** library. That formula:

$$Z = \frac{X - \mu}{\sigma}$$

Here,

X - Original value
μ - Mean of the column
σ - Standard Deviation of the column

3.1.5. DATA SPLITTING

Splitting the data is necessary for making a good accuracy. So, we need to split the input and output data to train and test with 90:10 ratio out of 100% using ***train_test_split*** function from ***model_selection*** in ***sklearn*** library. After splitting, I got four different dataset such as x_train, x_test, y_train, y_test.

3.1.6. DATA BALANCING

We need to balance the data for training set only to get a better accuracy in our model. If 80% input in our training data have one output value and remaining 20% input get another output value, it shows trainings are more familiar with one output and not more knowledge about another output. So, we need to balance the output training set (y_train) like 50:50 ratio with fake inputs using ***SMOTE*** function from ***over_sampling*** in ***imblearn*** library.

3.2. PREDICTION FOR NEW DATA

After prepared the dataset for apply the Machine Learning tools, we need to prepare a new row excluding output(y) with random data. After prepared the new row, we need to implement the row by using data preparation methods such as dividing, scaling, splitting and the add constant. After these all steps, the new input data was ready.

3.3. MODEL HYPERPARAMETER TUNING

In our model, we using ***GridsearchCV*** function from ***model_selection*** in **sklearn** library for hyperparameter tuning in Machine Learning Models such us ***Random Forest Classifier, Decision Tree Classifier*** and ***Support Vector Classifier.*** Here, we tuned the parameters of all the models and it will help to control the overfitting in the all classification methods.

For example in Random Forest Classifier model,

We used ***n_estimators*** parameter because it is the most fundamental hyperparameter in the ***Random Forest Machine Learning model.*** It have more efficiency and faster than other hyperparameters. We also take some other parameters that are ***max_featues="sqrt"*** and ***criterion="entropy"*** because these are perform good with classification models.

We tuned all models by using GridSearchCV with **5-fold Cross-Validation**. It helps our model to trained and tested **5 times** on different folds of the dataset. This process make the highest cross-validation accuracy was selected. After the all fitting and tuning, we got the best parameters and accuracy in all models.

3.4. EVALUATION METRIC

In our project, we selected the ACCURACY SCORE as an essential evaluation metric because it works well after applying ***SMOTE*** (Synthetic Minority Over-sampling Technique) in training set.

The overall correctness of the ML model was calculating the proportion of real predictions (both negative and positive) out of all prediction by accuracy score.

Since our prime focus was to maximize the correct classifications across the all classes and then after preprocessing, the class distribution was balanced. These all processes shown that the accuracy became a perfectible and suitable metric for my classification type problem.

According to our model, we got an good accuracy score in **Random Forest Classification** model that is near to 90 percentage (~ 89.94, after tuning). We balanced using **SMOTE** making good accuracy, however accuracy is sensitive for data imbalance.

CHAPTER - IV

MACHINE LEARNING MODELS

5.1. LOGISTIC REGRESSION

Logistic regression is a Supervised Machine Learning technique to perform the task of classification. This model comes under statistical model and It will help to give a solution for binary classification problems. It uses a sigmoid function to map an input to a probability between 0 and 1, predicting the likelihood of a sample belonging to a specific class. The output is a normalized probability, expressed as a value between **0 and 1**.

5.2. RANDOM FOREST CLASSIFICATION

Random Forests are one of the most common examples of ensemble learning. Actually **Random Forests** is Instead of building a single decision tree and use it to make predictions, build many slightly different trees and combine their predictions. ensemble learning uses majority vote to reduce variance. In this model, regularization hyperparameters such as **n_estimators** and **max_features** were help to avoid overfitting. It was usually works better than a single Decision Tree models because of an averaging.

5.3. DECISION TREE CLASSIFICATION

Decision tree learning is a machine learning method that employs a decision tree as a predictive model. A Decision Tree is a **tree-structured classifier** that splits the dataset based on attribute values. A decision (based on a feature) represented by each internal node, outcomes represented by branches and then final predictions represented by leaf nodes. We regularized it by using hyper parameters such as **max_depth** and **criterion(entropy)** because it can easily overfit.

5.4. SUPPORT VECTOR CLASSIFICATION

Support Vector Machine or ***SVM*** is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. The goal of the **SVM classifier** is to create the best line or decision boundary that can separate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. Finding the **best hyperplane** that separates data points of different classes with the **maximum margin** will help to run the model. It will handle non-linear decision boundaries by using the **kernel tricks** such as linear, polynomial, RBF and sigmoid. In this model, it will avoid the overfitting by using the hyperparameters like **C** (regularization strength) or **gamma** (influence of points in RBF kernel).

CHAPTER - V

OVERFITTING AVOIDANCE MECHANISM

4.1. HYPERPARAMETER SELECTION

According to Machine Learning models, overfitting is avoided by using regularization tools. According to ***Logistic Regression***, it did not have any special mechanism for avoid the overfitting.

Some regularization mechanisms in ***Random Forest Classifier***:

- ***max_depth*** - It helps to limit the maximum depth of each tree.
- ***max_features*** - This function works for adding randomness and reduce unnecessary variance.
- ***n_estimators*** - It helps to find the suitable number for generalize better from too many trees.

In our model, we used **max_features** and **n_estimators** mechanisms to avoid overfitting.

Regularization mechanisms in ***Decision Tree Classifier***:

- ***max_depth*** - It helps to control the maximum depth of the tree.
- ***min_samples_split*** - It reduces too many unwanted splits.
- ***min_samples_leaf*** - It helps to avoid making leaf nodes with few samples
- ***max_features*** - This function works for adding randomness and reduce unnecessary variance.
- ***criterion*** - It helps to choose best split that reduce entropy the most.

In our model, we used **max_depth** and **criterion** mechanisms to avoid overfitting.

Regularization mechanisms in ***Support Vector Classifier***:

- kernel - It helps to the dataset performed well at a higher dimension.
- epsilon - This parameter is used to make the good hyperplane.
- C (Regularization parameter) - It helps to covert the hard margin to soft margin , Controls the model complexity and prohibit overfitting by curbing high weights.

In our model, we used these all mechanisms to avoid overfitting.

4.2. CROSS VALIDATION

Cross validation is most powerful method for avoid the overfitting problems in all classification and regression type datasets. Especially, **5-fold Cross Validation** is the tuning method which is helps to separate our training and testing dataset into 5 folds and make a 5 times practice well. We apply this concept under ***GridSearchCV*** tuning method like *****CV=5*****.* This mechanisms helped us to avoid overfitting problem in our project.

4.3. FEATURE SELECTION

Feature selection is the crucial process for avoid overfitting and get a good accuracy. If our dataset have more features, it make a bad accuracy and overfitting or underfitting problems. So, we need to delete unnecessary features using some functions in python. According to our project, we used some essential functions to avoid overfitting problem and get a good accuracy. That functions are below:

- ***drop_first=True*** - After using ***get_dummies*** function from pandas library, we got more features according the different values of one column. If any column have 10 different values, we get 10 different column which have only numerical values(0 or 1) after using the ***get_dummies*** function. If we use the ***drop_first=True*** function, it will help to delete the first new column of every main old columns. According to theory of binary encoding, the value is not change however we delete the first column. So, we used this function for reduce our dataset by drop the unwanted features and avoid the overfitting problems.

- ***TargetEncoder*** - This function is most powerful function for encoding the column which have more than 10 different values. If we use ***drop_first=True*** function for a column which have more than 10 different values, we get minimum 10 different new columns after delete the first new column. So we installed the ***category_encoders*** library using **pip install** function for using the ***TargetEncoder*** function. According to this function, it helps to make a specific numerical values for each different values based on the values of target variable after divide the dataset into input and output(target). In our dataset, two column have 12 different values in each. If we use the ***drop_first=True*** function, we get 22 new columns from those 2 columns. So, we used ***TargetEncoder*** function and didn't get any new columns but datatype was changed to float from object.

CHAPTER - V

CONCLUSION

The ***Random Forest Classifier*** model gave the highest accuracy (89.94% after tuning) and the ***Decision Tree Classifier*** model was not performed well like ***RFC***.

Naturally, ***Random Forest*** model helps to avoid overfitting, performed well in both classification and regression, handle the non-linear relationships and perform well even if any variables are missing or bad feature selections. According to Support Vector Classifier, SVC suggested the ***Random Forest Classifier*** (criterion='entropy' & n_estimators=500) as a best model.

We can also prove that the best model by using confusion matrix.

Models	Predicted value = Actual value	Predicted value ≠ Actual value
Logistic Regression	374	79
Random Forest Classification	388	65
Decision Tree Classification	362	91
Support Vector Classification	391	62

According to this above data from confusion matrix of all models, Support Vector Classification predict more correct values than others. Although Support Vector Classification model was good, It suggested the **Random Forest Classification** as a **best model** for our project because of the reason maybe SVC have overfitting problems.

According to our experimental results, we would strongly recommend *Random Forest Classifier (RFC)* model for real-world predictions.

REFERENCES:

1. Hosmer, D.W., Lemeshow, S. and Sturdivant, R.X., 2013. *Applied Logistic Regression*. 3rd ed. Hoboken, NJ: Wiley.
2. Breiman, L., 2001. Random forests. *Machine Learning*, 45(1), pp.5–32.
3. Scikit-learn Developers (2025) *Logistic regression*. scikit-learn. Available at: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (Accessed: 25 August 2025).
4. Quinlan, J.R., 1986. Induction of decision trees. *Machine Learning*, 1(1), pp.81–106.
5. Cortes, C. and Vapnik, V., 1995. Support-vector networks. *Machine Learning*, 20(3), pp.273–297.
6. James, G., Witten, D., Hastie, T. and Tibshirani, R., 2021. *An Introduction to Statistical Learning with Applications in R and Python*. 2nd ed. New York: Springer.
7. Géron, A., 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2nd ed. Sebastopol, CA: O'Reilly Media.
8. Kuhn, M. and Johnson, K., 2013. *Applied Predictive Modeling*. New York: Springer.
9. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, pp.2825–2830.
10. Brownlee, J. (2020) *A gentle introduction to decision trees for machine learning*. Machine Learning Mastery. Available at: https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/ (Accessed: 25 August 2025).

* * * * * * * * *