# Machine Learning Meets Term Life Insurance: Targeting High-Value Customers

# 1. Introduction

## Overview of HashSysTech Insurance

HashSysTech Insurance is a forward-looking and growing insurance company specializing in the provision of insurance services, especially term life assurance. In the years past, this company has used various methods to market its products to various potential customers, and out of all these marketing strategies, the company has dramatically relied on telemarketing as one of the most effective means through which it can get its customers to buy its products. However, as competition increases and the cost of doing business increases, HashSysTech is faced with the future problem of achieving efficient resource utilization and consistently generating high returns from these campaigns. To this effect, the company is on the lookout for tools that will enable it to obtain relevant information that can be used for targeting prospects with high levels of potential.

## Project Greenlight and Term Life Insurance Targeting

To counter the effects of rising costs and the need to be efficient, HashSysTech embarked on a new project called **Project Greenlight**, which aimed at coming up with improved telemarketing techniques so as to pinpoint who among the population is likely to accept the offered term life insurance. The aim here is to lessen the impact of marketing and, at the same time, ensure that the campaigns are much more relevant, cheaper, and effective. By applying this project, HashSysTech shall reduce the chances of wastage of resources by only concentrating on those potential clients who are most likely to create a sale. The solution lies in having a robust and wealthy machine learning model that would help analyze parts of customer data and the possible following conversions, as well as other vital strategic decisions of the subsequent campaigns that need to be created. This assignment focuses on the creation of such a model to help the company improve its telemarketing business by increasing the effectiveness of its telemarketing calls made to potential clients.

# 2. Task 1: Data Exploration and Preparation

## 1.1 Data Exploration

Data exploration is the first step in the data analysis process. It allows one to get acquainted with a new set of data, especially its structure, patterns, and key statistics. Data exploration also helps the analyst recognize patterns, correlations, and outliers that may affect the model.

The data set we were privileged to work with is from HashSysTech Insurance Company and includes features about customer characteristics, communication, and past telemarketing sales experiences. This step involves summarizing numerical and categorical features and examining them in correlation to the variable that defines whether the customer transitions to term life insurance.

### Descriptive Statistics

For numerical features, measures of central tendency in the form of Mean and Median and measures of dispersion in Standard Deviation are conducted. These measures enable one to glimpse how the information is distributed and look for any irregularities in the data.

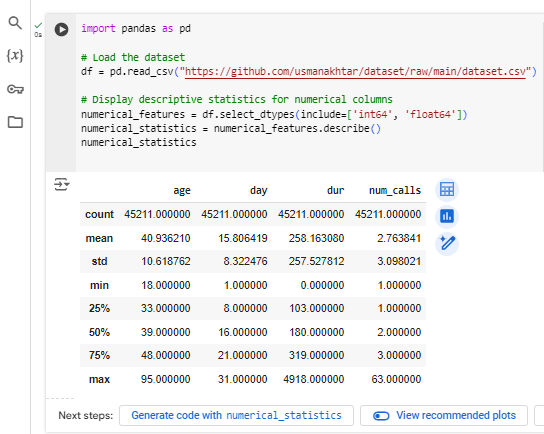


Figure 1: Python code for calculating descriptive statistics

This code calculates the count, mean, standard deviation, minimum, maximum, and quartile of the numerical features so that Analysts can get a complete picture of how these values are distributed in a dataset.

The frequency distribution of the category is also computed in the case of categorical features to identify the occurrence of the specific category. This step makes it easier to identify the common characteristics that define the majority of the customers.

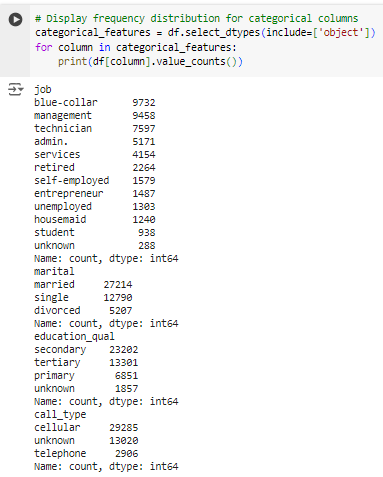


Figure 2: Python code for analyzing frequencies of categorical features

This process gives one an insight into the demographic attributes of these customers, such as age, gender, marital status, and other important classes.

### Correlation Analysis

Understanding how different features relate to the conversion is essential to explore their importance. A correlation matrix can be computed to check the degree of relation between each of the numerical variables and the target variable or any other numerical variable. If two exact features are strongly correlated, these features could significantly impact the model.

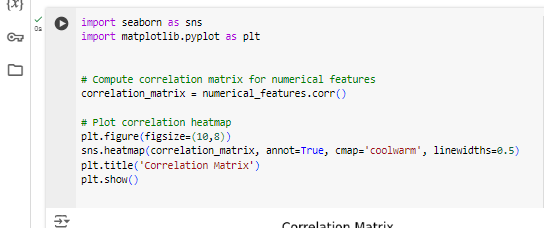


Figure 3: Python code to compute the correlation matrix

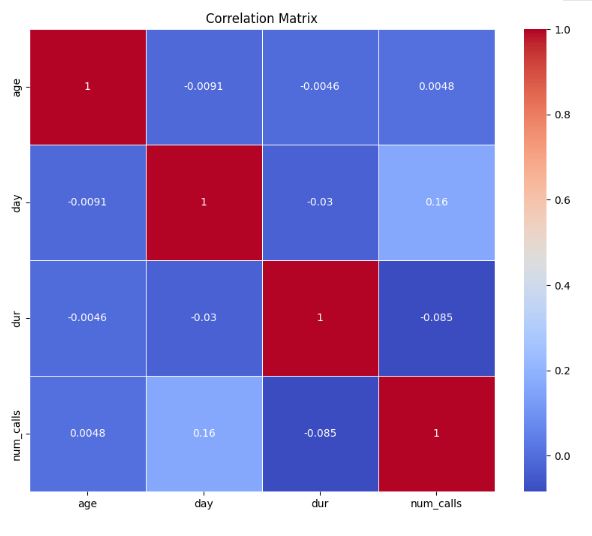


Figure 4: Correlation Matrix of Numerical Features

The heatmap helps determine which numerical features correlate strongly with the conversion variable, which dictates the next model and feature engineering.

## 1.2 Handling Missing Values and Outliers

Missing data and outliers are essential to handling data preparation as they affect the performance and accuracy of most machine learning algorithms.

### Handling Missing Values

The first step is to check the dataset to see if any missing values exist. Based on the proportion of missing data and the feature type, different techniques can be applied.

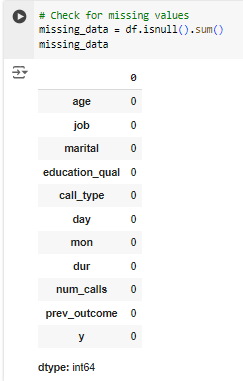


Figure 5: Python code to identify missing values

As seen from the above results, this dataset does not have missing values; therefore, no action is required to handle the missing data. This is beneficial because it ensures all features are available and can be used without further pre-processing steps.

Thus, it is essential to be aware of the several approaches to handling missing values in a database. Even though no missing values have been observed in the dataset of this particular case study, missing data is a phenomenon that is frequently encountered in actual databases. Inadequate data may bring out biases in the entire study; hence, such data must be handled well to avoid misleading conclusions.

#### Listwise Deletion (Removing Rows with Missing Data)

In case of missing data, you can remove the rows that contain missing values.

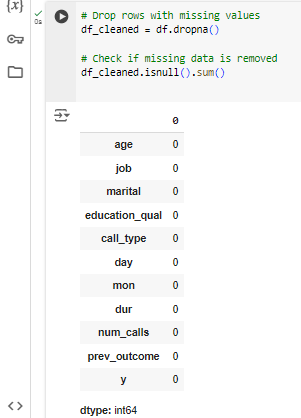


Figure 6: Python Code: Removing Rows with Missing Data

#### Imputation Techniques

If deletion is not the best approach, imputation is always an option. Below are some procedures that can be used to input the missing data.

##### Mean/Median Imputation for Numerical Data

Mean or median imputation is often used for numerical data, especially when the distribution is not highly skewed.



Figure 7: Python Code: Mean Imputation for Numerical Features

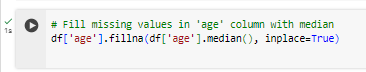


Figure 8: Python Code: Median Imputation for Numerical Features

##### Mode Imputation for Categorical Data

Categorical variables can be imputed using the most frequent value (mode).

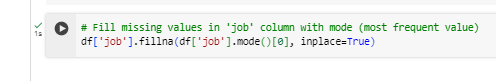


Figure 9: Python Code: Mode Imputation for Categorical Features

#### K-Nearest Neighbors (KNN) Imputation

The KNN imputation technique fills in the missing values based on the distance closest to it. This is useful, especially in cases of feature disparities and correlations.

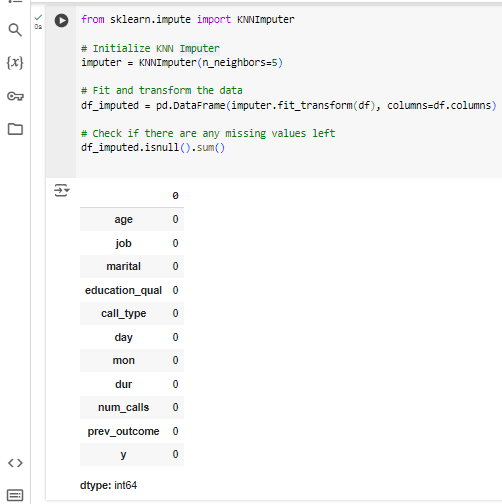


Figure 10: Python Code: K-Nearest Neighbors Imputation

#### Multiple Imputation with MICE (Multiple Imputation by Chained Equations)

MICE is a more powerful imputation approach that successively estimates missing observations and develops several data sets for the last analysis.

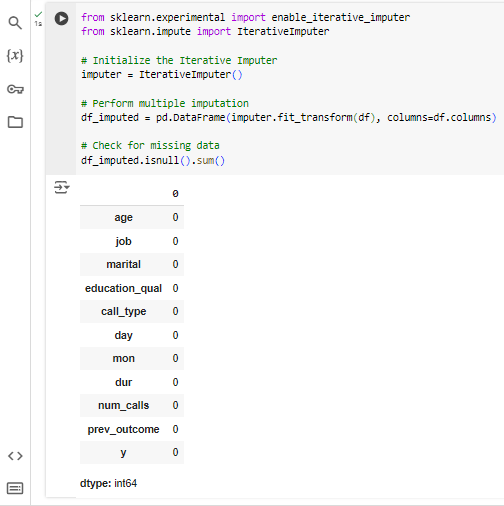


Figure : Python Code: MICE Imputation for Missing Values

#### Dropping Columns with Too Many Missing Values

Whenever some of the columns contain so many blank values, it is advisable to exclude such variables.

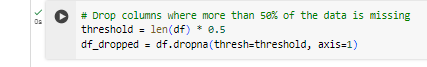


Figure : Python Code: Dropping Columns with Excessive Missing Data

It is noteworthy that missing data is critical in data analysis, and depending on the data type, the extent of missing data, and the goal of the analysis, it could be handled by deletion, imputation, or regression. These methods are helpful for future projects and are fundamental even in a dataset where they are not a prerequisite in the research process.

### Detecting and Handling Outliers

In statistics, outliers are extreme values that we can easily distinguish from other values in the database. They can potentially affect the output of a model, which is why, if present, they have to be identified properly.

In numerical features, outliers can be detected by observing the box plots or calculating the values' Z-scores. Data points with a Z-score greater than or equal to 3 can be examples of thresholds for outliers.

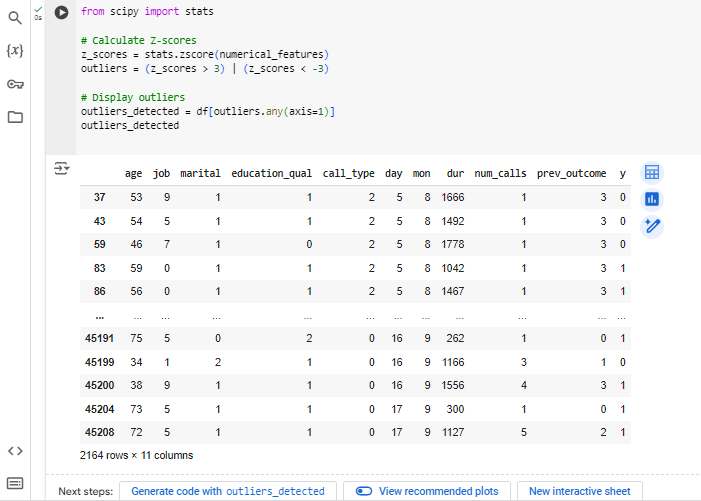


Figure : Python code to detect outliers using Z-scores

Once outliers are identified, they can be dealt with by employing techniques such as removing, transforming, or capping by Winsorization.

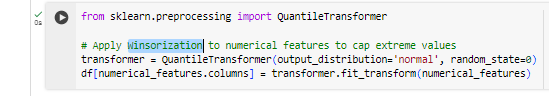


Figure : Python Code: capping outliers using Winsorization

This ensures that extreme values do not disproportionately influence the machine learning model.

## 1.3 Data Visualization

For exploratory purposes, graphing the dataset becomes necessary to see the correlation and trends between these features. This also assists in generating our hypothesis about which features might be significant in terms of term life insurance conversion.

### Scatter Plots

Scatter plots can be used to corroborate the impact of two quantitative variables and the target variable.

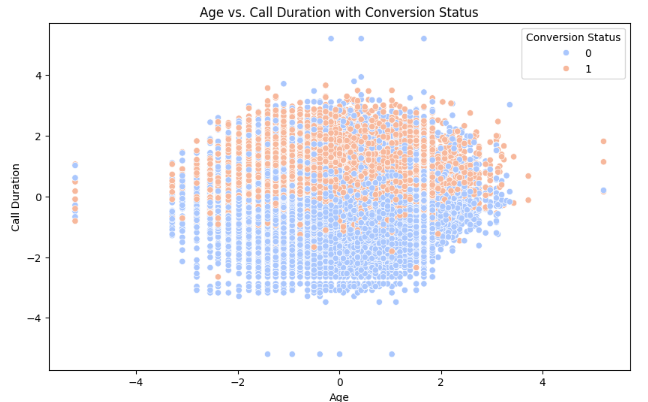


Figure : Scatter Plot: Age vs. Call Duration with Conversion Status

It can aid the investigation of the outcome by age or income to determine which category is more likely to convert in history so that relevant features can be selected.

### Histograms

Histograms are useful for examining the distribution of individual numerical features.

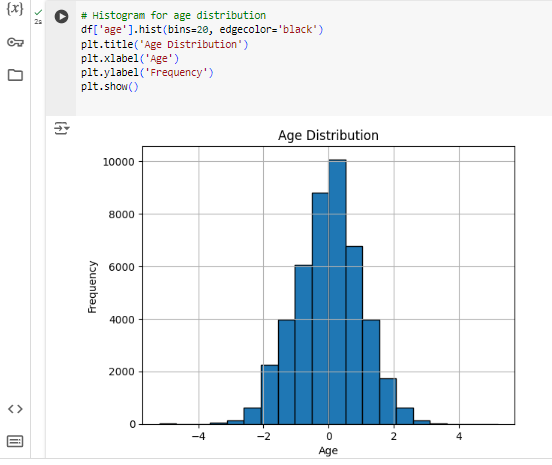


Figure : Python code for generating histograms

This offers information on the customers' age structure, and one can trace this to conversion patterns.

### Box Plots

Box plots are ideal for determining the presence of outliers and data distribution concerning the categories.

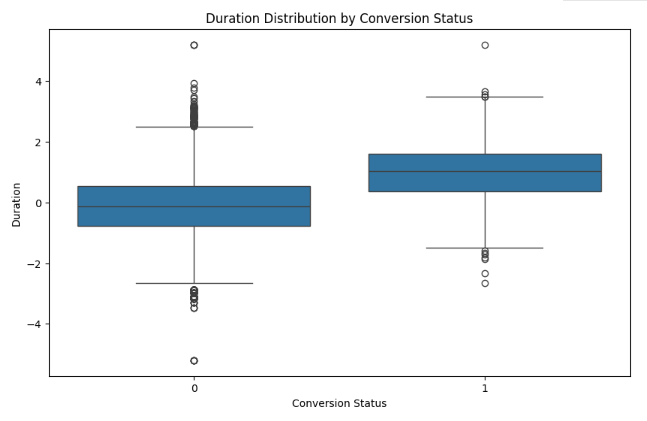


Figure : Box Plot: Duration Distribution by Conversion Status

This box plot then displays the income distribution of the customers who converted to the target and the customers who did not, allowing the viewer to quickly and easily pinpoint any distinctions. These visualizations, along with the summary statistics obtained in the previous step, allow for forming hypotheses for the tasks, which will be developed in the next step of the analysis.

To sum up, Task 1 provides the ground for constructing the strong machine learning model because it helps to ensure that the information is clear and clean and, therefore, clearly visualized. This is important in order to pave the way for the next steps, which include model selection, training, and the evaluation of the ideal model.

# 3. Task 2: Model Selection and Training

## 2.1 Model Selection

This is also essential in building a model solution because it defines the best algorithm for the data set and the problem to be solved with the data set. In this task, therefore, the aim is on the classification problem, which entails the variable of interest, y, being equal to 1 if a customer has converted or 0 if they haven’t. Several classification models are used, and among them, an optimal is chosen, which includes the aspect of the complexity of the model, the time required for training, accuracy attained, and the interpretability of the model.

* **Step 1**: Understanding the Problem. The aim of this task is to estimate whether a particular customer tends to convert, considering several features such as age, job, education, etc., and call the title back. Since it is a binary classification problem, a model is needed to help classify the problem into two results.
* **Step 2**: Evaluating Suitable Models For binary classification problems, the following models are commonly evaluated:

* Logistic Regression: This basic model is used to compare with other models to check performances. This method is applicable only if the relation between input features and target is linear.
* Decision Tree Classifier: An over-parameterized model that is interpretable and can fit decision surfaces of a high degree of complexity. Nevertheless, it is susceptible to overfitting.
* Random Forest Classifier: A combination of decision trees that can reduce over-learning and produce better products.
* Support Vector Machine (SVM): High-dimensional data concordance, but convergence may be an issue for big data.
* K-Nearest Neighbors (KNN): A concept that overlays data points by identifying a majority class in the neighboring data points. It depends on noise and other irrelevant features, for example.

After the models have been selected, they are assessed using cross-validation, in which the data set is partitioned into the training and validation data sets.

* **Step 3**: Selecting the Model. The Decision Tree Classifier is initially used in this scenario due to its interpretability and its ability to handle both nominal and quantitative variables. Moreover, two more models, random Forest and Logistic Regression models will also be implemented.

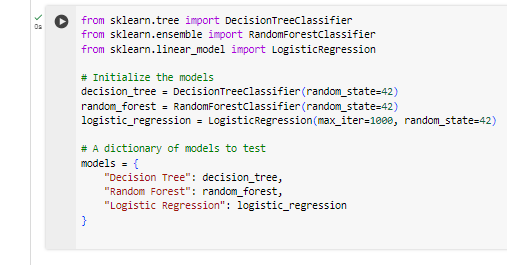


Figure : Jupyter Notebook Code: Model Initialization

## 2.2 Data Splitting

To implement the models, the data is first divided into training data and testing data. Training data is employed for training purposes, and testing data is used to test the model's performance.

* Step 1: Division Of Dataset The information is divided into training and testing independent datasets where the former is 80% while the latter is 20%. Some of the most common functions used for this purpose are the train\_test\_split from the Scikit-learn.

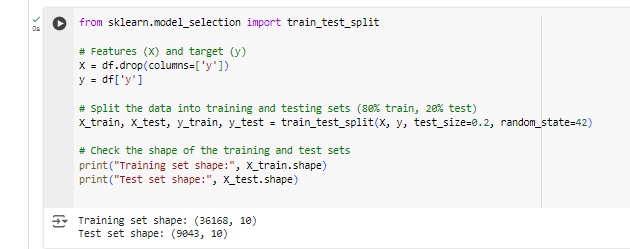


Figure : Python Code: Splitting the Dataset into Training and Testing Sets

In this step, categorical features, including job, marital, and education\_qual, might have to be feature-engineered for splitting, as most machine learning models process numeric inputs.

* Step 2: Feature Encoding with LabelEncoder. Label encoding is a transformation where features with nominal data types are transformed into integer types by replacing all features with different numbers. Since the dataset contains categorical features, such as job, marital, and education qualifications, the features will need to be encoded to train the models. LabelEncoder from sklearn has been utilized to transform each categorical predictor into numerics.

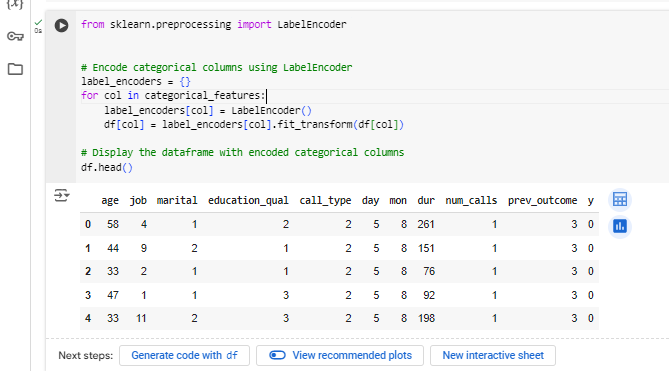


Figure : Python Code: Encoding Categorical Columns Using LabelEncoder

LabelEncoder assigns a numeric value as a label to each category in a specific column. For instance, assuming the job column contains the values management, technician, and blue-collar, the labels will be 0, 1, and 2. The dictionary label\_encoders stores a LabelEncoder object for each column so that the encoding can be decoded later if needed.

Hence, by applying LabelEncoder, the output dataset is converted into numeric form to fit machine learning models that only accept numerical inputs.

* Step 3: Scaling the Data. The features may eventually need to be scaled, especially for specific algorithms like logistic regression or KNN. Standardization is one of the most commonly used techniques where the values are adjusted for every feature such that its value has zero mean and unit variance.

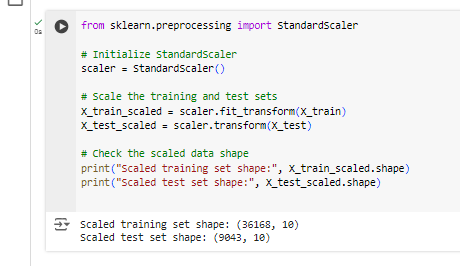


Figure : Python Code: Standardizing the Dataset

Once the data has been split and encoded, the models will be ready to train on the processed data set.

### Summary of Steps

In model selection, some of the algorithms relevant to binary classification were considered. Thus, the Decision Tree, Random Forest, and, at last, the Logistics Regression algorithm were selected for further investigation. In this work, data splitting was done to the extent of 80/20 to arrive at the training set of data points as well as the test set of data points. The categorical features were concretized by means of one hot encoder, and the necessary scaling was performed.

These preparations steps guarantee that the models shall go through properly formatted and separated data set which forms the basis to the subsequent process of training and evaluating the models on the data set in question.

# 4. Task 3: Model Interpretation and Evaluation

## 3.1 Model Interpretation

Interpretability in machine learning is the process by which the trained model can be explained based on its decision-making, which is the essential feature for the particular decision. For this task, the following steps will be performed:

* Train the Primary Model (Decision Tree)
* Evaluate and Compare Alternative Models (Logistic Regression and Random Forest)
* Feature Importance Analysis

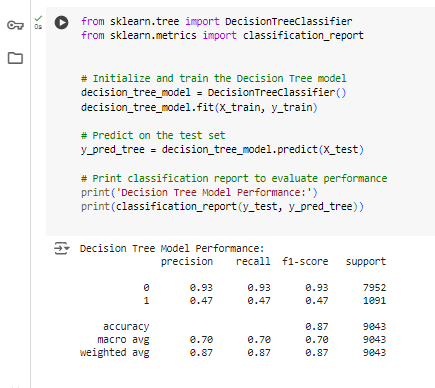


Figure : Python Code: Training and Evaluating the Decision Tree Model

### Decision Tree Model Interpretation

The Decision Tree model is assessed concerning its capacity to categorize an instance into the given target classes. The model's performance metrics are as follows:

* Accuracy: **0.8712**
* Precision: **0.4667**
* Recall: **0.4748**
* F1 Score: **0.4707**

#### Key Insights:

* Accuracy: It accurately classifies approximately 87.12% of the instances, showing high overall correctness.
* Precision: The precision of the model for the positive class (conversion) is 46.67%, which is the amount by which the model accurately identified a positive instance among the total number of predicted positives.
* Recall: The recall for the positive class is 47.48%, meaning the model can control 47.48% of the actual positive cases.
* F1 Score: This means that the model achieved a 47.07% F1 score, which shows that the model has both precision and recall rates. The Decision Tree exhibits satisfactory results, underscoring a rather good accuracy of the positive cases, but it may be fine-tuned.

#### Feature Importance from Decision Tree

The Decision Tree model also showed feature importance, where the model identified which features contributed most to arriving at the decision. The following table below gives an overview of the feature importance values.

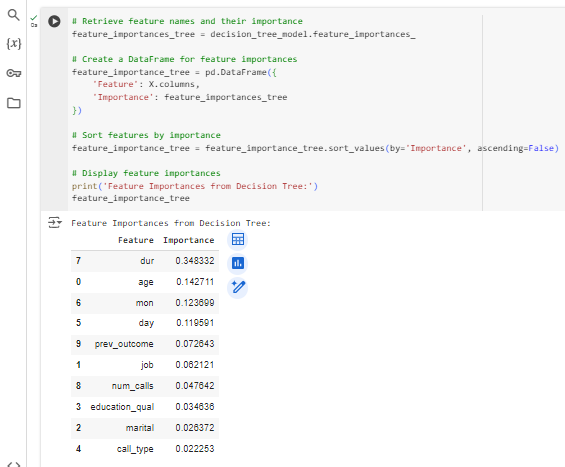


Figure : Python Code: Feature Importance from Decision Tree

|  |  |
| --- | --- |
| **Feature** | **Importance** |
| Duration (dur) | 0.3483 |
| Age | 0.1427 |
| Month(mon) | 0.1237 |
| Day | 0.1196 |
| Previous Outcome | 0.0726 |
| job | 0.0621 |
| Number of calls | 0.0476 |
| Education Level | 0.0346 |
| Marital Status | 0.0264 |
| Call type | 0.0223 |

#### Key Insights:

The length of a call proves most pivotal in the likelihood of each caller’s conversion, with age and month as the following relevant predictors. Other variables such as job, education level and marital status as well as others also affect the model predictions’ but to a lesser extent.

### Logistic Regression Interpretation

The Logistic Regression model provides a different perspective on classification performance:

* Accuracy: **0.8872**
* Precision: **0.5978**
* Recall**: 0.1989**
* F1 Score: **0.2985**

#### Key Insights:

The classification of Logistic Regression is 88.72% in general, and the precision of the positive class is 59.78%. Nevertheless, it has a recall of only 19.89%, meaning it gets very many of the actual positive cases wrong. This results in an F1 score of 29.85%, which means that the model was not as powerful in identifying positive cases as it was in other cases, and there is a compromise between precision and recall.

### Random Forest Interpretation

The Random Forest model provides a robust performance metric:

* Accuracy: **0.9040**
* Precision: **0.6509**
* Recall: **0.4409**
* F1 Score: **0.5257**

#### Key Insights:

The findings reveal that the Random Forest model is superior to all the classification models with a high accuracy of 90.40%, precision of 65.09% and recall of 44.09% for the positive examples. This is seen in the fact that its F1 score stands at 52.57%, which shows high accuracy in the balance between Precision and Recall, thus suitable for classification as compared to the other metrics.

## 3.2 Model Evaluation

The evaluation metrics for each model are summarized in the table below, providing a clear comparison of performance:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| Decision Tree | 0.8712 | 0.4667 | 0.4748 | 0.4707 |
| Logistic Regression | 0.8872 | 0.5978 | 0.1989 | 0.2985 |
| Random Forest | 0.9040 | 0.6509 | 0.4409 | 0.5257 |

* Decision Tree: That said, the model gives reasonable accuracy and feature importance, which can be a downside to its lower precisions and recalls, particularly for the positive class, making it perhaps not the most suited model for high-risk classification.
* Logistic Regression: Very accurate and precise, but only identifies a few true positives and, therefore, has low recall.
* Random Forest: A type that demonstrates excellent performance in terms of accuracy, precision, and recovery balance of recall. It is the most accurate model for forecasting conversion of the three models.

The proposed approach is based on the Random Forest model, which performs the best on all metrics. We could achieve an even higher improvement of the given results due to the additional tuning and validation of the model.

# 5. Conclusion

## Summary of Findings

Three classification models, Decision Tree, Logistic Regression, and Random Forest, were used to make customer conversion predictions for HashSysTech Insurance. The Decision Tree model yielded fair accuracy in terms of both precision and recall, and the best predictor was the call duration. Logistic regression yielded 88.72% accuracy and 59.78% precision, and there was relatively low recall, but positive case identification was also low. Random Forest had the highest accuracy of 90.40% while having a low precision of 65.09% and recall of 44.09% compared to the other models.

## Business Implications for HashSysTech Insurance

The results of the model evaluations provide significant implications for HashSysTech Insurance's marketing strategies and decision-making processes:

* Targeted Marketing Efforts: Customer conversion can be improved using the Random Forest model because it predicts better than the marketing campaign of the other two model campaigns, which target people with a high chance of being converted with the help of HashSysTech Insurance.
* Efficient Use of Resources: This means that marketing costs can be efficiently managed by focusing on high-value/ potential leads and avoiding or minimizing spending on low-value/potential leads. Hence, the chances of conversion and overall ROI are enhanced.
* Strategic Insights: The Decision Tree model helps understand how customers respond to HashSysTech Insurance engagement, such as call duration.
* Ongoing Optimization: It is recommended that HashSysTech Insurance incorporate the workflows into the models and continuously update them to meet the ever-changing customers’ behaviours and markets.

Therefore, interpreting outcomes obtained from the Random Forest model and integrating these outcomes into the company’s marketing strategies will enable HashSysTech Insurance to make better decisions about resource allocation and enhance the company’s performance.